

# **Uncertainty in Deep Learning: A Probabilistic Robotics Perspective**

Zur Erlangung des akademischen Grades eines  
Doktors der Naturwissenschaften

von der KIT-Fakultät für Informatik  
des Karlsruher Instituts für Technologie (KIT)

genehmigte  
Dissertation  
von  
Jongseok Lee

Tag der mündlichen Prüfung: 30. Oktober 2025

1. Referent: Prof. Dr. rer. nat. habil. Rudolph Triebel
2. Referent: Prof. Dr. rer. nat. Marc Toussaint
3. Betreuer: Prof. Dr.-Ing. Tamim Asfour





---

## Acknowledgments

---

This dissertation is based on research I conducted while working at the Institute of Robotics and Mechatronics of the German Aerospace Center (DLR) as a research scientist. During the last five years, I had the fortune to be also a PhD candidate at the Karlsruhe Institute of Technology (KIT), which led to finalizing this thesis. While pursuing a PhD was definitely not easy, I had great people around me, who were not only among the best robotics researchers world-wide, but who were caring colleagues, willing to give me more than what I can give them back. So, I take space here to most sincerely express my gratitude.

First and foremost, I am thankful to my PhD advisor, Prof. Rudolph Triebel, who is also my department head at DLR and also a mentor. Back when I started at DLR, I recall many open discussions within the department. A senior scientist argued, “DLR is about cool robots, not fancy publications”. And I remember Rudolph trying to convince people that scientific contributions are also important, and these contributions will only make the DLR robots even cooler. In retrospect, Rudolph has shown that he was right. Under Rudolph’s leadership, our department’s scientific output grew, many colleagues have won both academic and industrial awards, and he is now a professor at one of the major technical universities in Europe. At the same time, all the DLR robots now have more advanced perception and cognition capabilities, and some colleagues are even the main contributors of important space missions, like the MMX project, where a rover is being sent to a moon of Mars. Being part of such a dynamically growing and high-performing department enabled me to generate results that I am personally proud of. I shall also not forget many of Rudolph’s advices. One of them was, roughly, “I survived my PhD because I collaborated with others. It should not be that different for you”. Thanks to this advice, I ended up surviving the devastating competitions in the current AI research and had fun working with so many colleagues, which forms the best memories of my PhD experience. For these reasons and many more, I thank Rudolph for the mentorship over so many years.

As important, I express my sincere gratitude to my PhD advisor, Prof. Tamim Asfour. Back in 2020, I remember the excitement when Tamim gave me the opportunity to pursue my research as his external PhD student. In his first email, Tamim wrote “You will be as any of my PhD students”. And this happened to be true although I was at DLR, not in Karlsruhe. Tamim invited me to many of the annual retreat events and I could also spend about half a year as a guest researcher in his laboratory. These experiences have been truly amazing, enabling me to expose myself to different ideas, research methods, and robotic innovations. So, once again, I would like to thank Tamim for all the opportunities.

---

Prof. Marc Toussaint from TU Berlin has been kind to serve as an external reviewer of this thesis. It's truly a big honor for me that my dissertation gets reviewed by the renowned scientist in the area of probabilistic inference and more. I am also grateful to my institute director, Prof. Alin Albu-Schaeffer, for hosting me at his institute at DLR for many years.

My life at DLR started when Dr. habil. Konstantin Kondak accepted to supervise my master thesis. Since then, Konstantin has been my role model – a scientist who genuinely believes in the importance of bringing research findings from academia to industry, and an experienced and highly capable engineer who can innovate and develop practical solutions to challenging problems. Looking back, Konstantin put trust in me and gave me so many opportunities for growth. For example, during my hiring process at DLR as an employee, I had interviews with Rudolph at TU Munich because Rudolph had lectures on that day. To meet Rudolph, Konstantin drove all the way from DLR to TU Munich, persuading Rudolph that I can fit the position (despite my studies being in control theory while the open position was on computer vision). At the later stage of my PhD, Konstantin also gave me opportunities to lead his group on his behalf for our participation in a robotics competition. Together, we also went through the DLR's start-up programs, seeking for potentials of technology transfer of our robotics research. All these experiences have changed how I think about research and what I want to achieve in my career. For these reasons and many more, I sincerely thank Konstantin for the mentorship and our long-lasting relationship.

They say, it takes a village to raise a PhD student. My wholehearted acknowledgments go to many colleagues that I worked with in the past.

To *"Uncertainty People."* I thank my close collaborators within our department: Jianxiang Feng, Matthias Humt, and also Dominik Schnaus. Without them, I would not have survived the lunatic times of the current AI research. Also, my best times at DLR were definitely with Jianxiang and Matthias, discussing many exciting ideas on uncertainty. Here, I would also thank the collaborators outside our institute, especially colleagues at other DLR institutes, and the PhD students across different universities with whom we had the pleasure of hosting the workshops on probabilistic robotics at conferences. Thinking about the fun moments with Jianxiang and Matthias at the workshop in Japan, it just brings me smiles.

To *"Team Spirit"*. I thank my beloved teammates: Ribin Balachandran, Simran Singh, Jianxiang Feng, Hrishik Mishra, Marco De Stefano, and Konstantin Kondak. Here, my special acknowledgment goes to Ribin. Without Ribin, I would not have survived not only my PhD studies but also life in general. I also learned so much from him, from how to collaborate and appreciate other colleagues to how we communicate our research. So many good memories with Ribin! From panic pushing for the AEROARMS project, many outdoor experiments with the robot SAM (with mosquitoes in summer and cold in winter), intense preparations for demonstrations at industrial exhibitions, to many business trips. I also thank the former members of the flying robotics group: Yura Sarkisov, Andre Coelho, and Min Jun Kim, who enormously contributed to the robot and shared many fun moments.

To *"Humanoids"*. Many appreciations to the wonderful colleagues at the High Performance Humanoid Technologies (H2T) laboratory at KIT. In particular, I thank Timo Birr for collaboration and support on ARMAR-6, Franziska Krebs for being a friendly office mate and support in integrating me into the lab, Byungchul An for approaching me first during the retreat, and Noemie Jaquier for hangouts at many conferences. Special thanks to the DLR TORO team: Prof. Jinoh Lee, George Mesesan, Robert Schuller, Konrad Fruend, and Samuel Bustamante. Automatica 2025 was truly amazing. Also, it has been a big honor to work on the DLR's flagship robot, TORO. I am especially grateful to Jinoh, who is also a source of inspiration. Jinoh is the best senior scientist I know when it comes to "working

---

with people”. He also has this positive and optimistic vibe, always so contagious.

*To “Aliens.”* ARCHES mission in Sicily was an amazing and unique experience. It was very inspiring to watch how a team of German engineers got the job done (under the leadership of Armin Wedler). Among mobile robotics groups, I thank my x-rotor teammates: Marcus Mueller, Florian Steidle, Nidish Raj, Moritz Kuhne, Wolfgang Stuezl and others. Special acknowledgment also goes to my esteemed current and former managers: Marcus Mueller, Wolfgang Stuezl and Maximilian Durner, who granted me the freedom to pursue what I am passionate about. I also thank Riccardo Giubilato for recent collaborations and fun times in attending conferences. Same thanks go to Arjun Vayugundla. From the x-rotor team, I also thank Julius Quell for being a nice office mate and a legend.

*To “Team Yantra”.* My appreciation goes to the proud Yantra team: Hrishik Mishra, Thomas Eiband, and Korbinian Nottensteiner. Hannover Messe 2024 was amazing – not only nice robots, but cool parties and tornadoes. Looking back, it could have been stressful when the deadlines were approaching. But, somehow, we ended up having more fun and delivering what we promised. I guess experience matters! I am especially grateful to Hrishik, with whom I had many fun times working together. Hrishik is a true engineer who is always occupied with robots and manifolds, which is a source of inspiration for me.

*To all my students.* Heartfelt acknowledgment goes to my students for their contributions (alphabetic order): Baptiste Lardinoit, Christian David Echeverry Valencia, Dominik Schaus, Hojune Kim, Johannes Gaus, Jyotirmaya Patra, Kashmira Shinde, Leonhard Chen, Manuel Schaus, Manuel Westermann, Matthias Humt, Omar Hedeya, Patrick Kroemer, Seok Jun Kim, Youngmin Song, Zhang Kai, and Ziyuan Qin. I am so proud of them.

I thank the proofreaders of this thesis: Ribin Balachandran, Matthias Humt, Klaus Strobl, Maximilian Durner, Ulrich Hillenbrand, and Riccardo Giubilato. Having been proofread by the experienced senior scientists at DLR gives me more confidence about my thesis.

Thanks to many amazing people, I could keep my sanity intact. My acknowledgment goes to Simran Singh, Arjun Vayugundla and Michi Panzirsch for many “discussions” downstairs, Tin Muskardin for being a mentor and showing me how to be social, and the gang at the DLR football club for the best hours of the week. Again, thanks to Ribin Balachandran, Simran Singh, Arjun Vayugundla and Hrishik Mishra for many get-together (regardless of my Indian caste). My best appreciations go to my older friends. I am thankful to Seonghun and Jihwan. They also did their PhD and often we could share similar challenges and joys. I also thank Imrul for the friendship. In the future, I hope to see all of you more often!

Most importantly, I would like to thank my family: my parents, my grandparents, and three sisters for their unconditional love and support. I have lived abroad for so many years, far away from home. But I always knew that my family is behind me. So, I could live a wonderful life that is filled with so many exciting adventures. With this opportunity, I want to express my most sincere appreciation and all the love I have for them!

Munich, September 2025

Jongseok Lee



Robots are physical systems that perceive, plan, and act in the real world. As a consequence, their mistakes can not only cause failures in the robots' mission, but they can even endanger human lives, in the case of a robotic surgeon or a self-driving car, for example. This motivates *probabilistic robotics*, i.e., a paradigm of robotics with a set of methods that enable the robots to assess the *uncertainty* in their sensory data, used algorithms, learned predictors, etc., such that the robots can plan safe actions. One of the challenges herein is *uncertainty quantification* in the systems that rely on neural networks. For this, Bayesian statistics provide theoretical foundations. However, bringing Bayesian statistics to neural networks – referred to as Bayesian Deep Learning – involves the problem of (a) the choice of well-specified priors, (b) the inference of the posteriors, and (c) the uncertainty estimation through marginalization, which are active areas of research in machine learning and beyond.

For all these sub-problems of Bayesian Deep Learning, this work provides novel methodologies that are well-suited for their applications to robotics. Concretely, we advance the generalization of learning algorithms through priors, the scalability of the inference algorithms to obtain complex posteriors, and the run-time efficiency of marginalization for predictions. We achieve this while improving the quality of uncertainty estimates when compared to existing methods. Inspirations have been drawn from the theories of generalization, information, as well as the universal approximation theorem of neural networks. Theoretical foundations are further provided for all the devised methodologies. With these results, we finally develop probabilistic robotic systems that can improve the performance of deep learning in real-world applications. Starting from (a) a humanoid robot recognizing deformable objects, (b) in-flight aerodynamic analysis of stratospheric and manned helicopter flights, (c) semi-autonomous aerial manipulation at night, to (d) shared autonomy with haptic and extended reality, we provide several system-level contributions.

As a result, the contribution of this thesis stands on two pillars of probabilistic robotics – one on methodological advances to quantify uncertainty, and another on systems contributions that show possibilities for new applications. Through this lens of probabilistic robotics, we provide a new perspective on the general problem of uncertainty in deep learning.

**Keywords:** Uncertainty, Probabilistic robotics, Aerial manipulation, Field robotics, Robot perception, Active learning, Shared autonomy, Aerodynamics, Humanoids.



<b>Acronyms</b>	<b>15</b>
<b>List of Symbols</b>	<b>17</b>
<b>I. Prologue</b>	<b>19</b>
<b>1. Introduction</b>	<b>21</b>
1.1. Probabilistic Robotics . . . . .	23
1.2. Problem Statement . . . . .	25
1.3. Contributions . . . . .	27
1.3.1. Main Contributions . . . . .	27
1.3.2. System-level Evaluation and Extensions . . . . .	28
1.4. Outline . . . . .	31
<b>2. Foundations &amp; Challenges</b>	<b>33</b>
2.1. Uncertainty in Robotics & Deep Learning . . . . .	33
2.2. Bayesian Statistics . . . . .	35
2.3. Approximate Bayesian Inference . . . . .	38
2.3.1. Linear Gauss Systems . . . . .	39
2.3.2. Bayesian Linear Models . . . . .	40
2.3.3. Bayesian Neural Networks . . . . .	42
2.3.4. Challenges Revisited . . . . .	44
2.4. Bayesian Deep Learning: A Brief Review . . . . .	46
2.4.1. On Priors in Bayesian Deep Learning . . . . .	46
2.4.2. On Posteriors in Bayesian Deep Learning . . . . .	47
2.4.3. On Predictions in Bayesian Deep Learning . . . . .	48
2.5. Benchmarks and Evaluation Protocols . . . . .	49
2.6. Summary . . . . .	52

<b>II. Bayesian Algorithms for Deep Neural Networks</b>	<b>53</b>
<b>3. On Prior Specification for Bayesian Neural Networks</b>	<b>55</b>
3.1. Introduction . . . . .	55
3.2. Learning Expressive Priors . . . . .	56
3.2.1. Notation and Background . . . . .	56
3.2.2. Prior Learning with Sums-of-Kronecker-Product Computations . . . . .	57
3.2.3. Derivations and Optimizations over Tractable PAC-Bayes Bounds . . . . .	59
3.2.4. Bayesian Progressive Neural Networks . . . . .	61
3.3. Related Work . . . . .	62
3.4. Experiments and Evaluations . . . . .	64
3.4.1. Ablation Studies . . . . .	64
3.4.2. Generalization in Bayesian Continual Learning . . . . .	66
3.4.3. Few-Shot Generalization and Uncertainty . . . . .	66
3.5. Summary . . . . .	68
<b>4. On Posterior Inference for Bayesian Neural Networks</b>	<b>69</b>
4.1. Introduction . . . . .	69
4.2. Posterior Inference in Sparse Information Form . . . . .	71
4.2.1. Notation and Background . . . . .	71
4.2.2. Approximate Inference in Information Form . . . . .	71
4.2.3. Model Uncertainty in Sparse Information Form . . . . .	73
4.2.4. Low-Rank Sampling Computations . . . . .	75
4.3. Theoretical Analysis and Guarantees . . . . .	76
4.4. Related Work . . . . .	77
4.5. Experiments and Evaluations . . . . .	78
4.5.1. Small Scale Experiments . . . . .	79
4.5.2. Active Learning . . . . .	80
4.5.3. Classification Tasks . . . . .	81
4.5.4. Large Scale Experiments . . . . .	82
4.6. Summary . . . . .	83
<b>5. On Predictive Distributions of Bayesian Neural Networks</b>	<b>85</b>
5.1. Introduction . . . . .	85
5.2. Predictive Uncertainty via Sparse Gaussian Processes . . . . .	86
5.2.1. Main Idea on Sampling-free Uncertainty Estimation . . . . .	87
5.2.2. Neural Networks as Sparse Gaussian Processes . . . . .	89
5.2.3. A Practical Learning Algorithm . . . . .	90
5.3. Related Work . . . . .	92
5.4. Experiments and Evaluations . . . . .	93
5.4.1. Toy Regression . . . . .	93
5.4.2. Learning Inverse Dynamics . . . . .	94
5.4.3. Probabilistic Object Detection . . . . .	95
5.5. Summary . . . . .	96



<b>III. Extensions and Applications to Robotics</b>	<b>97</b>
<b>6. Stream-based Active Learning for Robust Semantic Scene Analysis</b>	<b>99</b>
6.1. Motivation . . . . .	99
6.2. Related Work . . . . .	100
6.3. System Concept and Challenges . . . . .	102
6.4. CLEVER: A Stream-based Active Learner . . . . .	102
6.4.1. Underlying Prediction Model for Continual Adaptation . . . . .	102
6.4.2. Bayesian Learning for Uncertainty and Generalization . . . . .	104
6.4.3. A Temporal Active Learning System with Humans . . . . .	106
6.4.4. System Overview and Implementation Details . . . . .	107
6.5. Experiments and Evaluations . . . . .	107
6.5.1. Ablation Studies and Comparative Assessments . . . . .	107
6.5.2. A Complete Open-set Evaluation with Users . . . . .	109
6.5.3. Demonstration on a Humanoid Robot . . . . .	110
6.6. Summary . . . . .	111
<b>7. Learning Fluid Flow Visualizations from In-Flight Images with Tufts</b>	<b>113</b>
7.1. Motivation . . . . .	113
7.2. Related Work . . . . .	115
7.3. Data, Task, and Challenges . . . . .	116
7.4. Probabilistic Tuft Segmentation Pipeline . . . . .	117
7.4.1. Tuft Detection . . . . .	117
7.4.2. Tuft Classification . . . . .	119
7.4.3. Tuft Segmentation . . . . .	120
7.5. Experiments and Evaluations . . . . .	121
7.5.1. Results on Tuft Detection . . . . .	121
7.5.2. Results on Tuft Classification . . . . .	123
7.5.3. Results on Tuft Segmentation . . . . .	123
7.5.4. Results on Final Performance . . . . .	124
7.6. Summary . . . . .	125
<b>8. Realizing Telepresence Robots with Aerial Manipulation Capabilities</b>	<b>127</b>
8.1. Motivation . . . . .	127
8.2. Related Work . . . . .	130
8.3. System Description, Problem Statement and Challenges . . . . .	132
8.3.1. System Description . . . . .	133
8.3.2. Problem Formulation and Identified Challenges . . . . .	134
8.4. Introspective Perception Pipeline . . . . .	137
8.4.1. The Proposed Pipeline for Objects of Known Geometry . . . . .	137
8.4.2. The Proposed Pipeline for Objects of Unknown Geometry . . . . .	140
8.4.3. The Proposed Active Learning Framework . . . . .	143
8.5. Experiments and Evaluations . . . . .	148
8.5.1. Ablation Studies and Evaluations . . . . .	148
8.5.2. Field Testing and User Validation . . . . .	155
8.5.3. Discussion . . . . .	158
8.6. Lessons Learned . . . . .	159
8.7. Summary . . . . .	161

<b>9. Perceptive Shared Autonomy for Aerial Manipulation under Uncertainty</b>	<b>163</b>
9.1. Motivation . . . . .	163
9.2. Related Work . . . . .	164
9.3. Perceptive Shared Autonomy under Uncertainty in DL . . . . .	165
9.3.1. The Design of SPIRIT - Shared Autonomy Concept . . . . .	166
9.3.2. User Interfaces of SPIRIT . . . . .	167
9.4. Uncertainty-Aware Perception in SPIRIT . . . . .	168
9.4.1. Learning to Register Point Clouds with Digital Twins . . . . .	168
9.4.2. Uncertainty Estimation with Gaussian Processes . . . . .	169
9.5. Experiments and Evaluations . . . . .	171
9.5.1. Ablation Studies on SPIRIT's Perception . . . . .	171
9.5.2. User Studies on SPIRIT's Shared Autonomy . . . . .	172
9.5.3. Completion of Industrial Scenarios with SPIRIT . . . . .	174
9.6. Summary . . . . .	174
 <b>IV. Epilogue</b>	 <b>177</b>
<b>10. Applied Probabilistic Robotics</b>	<b>179</b>
10.1. Yantra: A Robot Apprentice for Textile Manufacturing . . . . .	179
10.2. The ARCHES Analog Space Demonstration Mission . . . . .	181
10.3. Perceptive Locomotion with the DLR TORO . . . . .	184
10.4. Paula: Post-hoc Probabilistic Programming . . . . .	187
10.4.1. Why Post-hoc Probabilistic Programming? . . . . .	188
10.4.2. Concept Description and Key APIs . . . . .	188
10.5. Summary . . . . .	190
<b>11. Conclusion</b>	<b>191</b>
11.1. Contributions . . . . .	191
11.2. Outlook . . . . .	194
 <b>V. Supplementary Materials</b>	 <b>197</b>
<b>A. Chapter 4 Appendix</b>	<b>199</b>
A.1. Preliminaries . . . . .	199
A.1.1. Fisher Information Matrix . . . . .	199
A.1.2. PAC-Bayesian Bounds . . . . .	201
A.2. Algorithmic Overview, Complexity and Extensions . . . . .	202
A.2.1. Progressive Bayesian Neural Networks . . . . .	202
A.2.2. Complexity of the Power Method for Sums of Kronecker Products . . . . .	205
A.3. Derivations and Proofs . . . . .	205
A.3.1. Derivation of the PAC-Bayes Objectives . . . . .	205
A.3.2. Proof of Lemma 3.2.1 . . . . .	211
A.3.3. Proof of Lemma 3.2.2 . . . . .	211
A.4. Some Thoughts on the Regression . . . . .	212
A.5. Implementation Details . . . . .	214
A.5.1. Implementation Details: Section 3.4.1 . . . . .	214
A.5.2. Implementation Details: Section 3.4.2 . . . . .	215

A.5.3. Implementation Details: Section 3.4.3 . . . . .	216
A.6. Additional Experiments and Results . . . . .	216
A.6.1. Qualitative Analysis of the Approximate Upper Bound . . . . .	216
A.6.2. Ablations in the Small-Data Regime . . . . .	216
A.6.3. Further Cold Posterior Experiments . . . . .	220
A.6.4. Few-Shot Learning: Results per Dataset . . . . .	221
<b>B. Chapter 5 Appendix . . . . .</b>	<b>223</b>
B.1. Derivations and Proofs . . . . .	223
B.1.1. Derivations of Low-Rank Sampling Computations . . . . .	223
B.1.2. Proof of Lemma 4.3.1 . . . . .	227
B.1.3. Proof of Lemma 4.3.3 . . . . .	229
B.1.4. Proof of Lemma 4.3.4 . . . . .	229
B.1.5. Proof of Lemma 4.3.5 . . . . .	230
B.2. Implementation Details . . . . .	230
B.2.1. Implementation Details: Section 4.5.1 . . . . .	231
B.2.2. Implementation Details: Section 4.5.2 . . . . .	231
B.2.3. Implementation Details: Section 4.5.3 . . . . .	232
B.2.4. Implementation Details: Section 4.5.4 . . . . .	233
B.3. Additional Experiments and Results . . . . .	233
B.3.1. UCI Benchmarks . . . . .	233
B.3.2. Spectral Sparsity of Information Matrix . . . . .	235
B.3.3. Effects of Low Rank Approximation . . . . .	235
B.3.4. Additional Results on Toy Regression . . . . .	236
B.3.5. Effects of Hyperparameters - UCI benchmark . . . . .	237
B.3.6. Additional ImageNet Results . . . . .	239
<b>C. Chapter 6 Appendix . . . . .</b>	<b>241</b>
C.1. Derivations and Proofs . . . . .	241
C.1.1. Derivations of Neural Networks and Local GPs Connections . . . . .	241
C.1.2. Derivations of Covariances per Loss Functions . . . . .	246
C.1.3. Proof of Lemma C.1.3 . . . . .	248
C.1.4. Proof of Lemma C.1.7 . . . . .	251
C.2. Discussion on Negative Log Likelihood Calculation . . . . .	252
C.3. Implementation Details and Additional Results . . . . .	253
C.3.1. Implementation Details for Laplace Approximation Variants . . . . .	253
C.3.2. Implementation Details for MC-Dropout Variants . . . . .	253
C.3.3. Implementation Details for Deep Ensembles . . . . .	254
C.3.4. Implementation Details and Additional Results for Local GPs . . . . .	254
<b>D. List of Publications . . . . .</b>	<b>259</b>
<b>Bibliography . . . . .</b>	<b>263</b>



---

## Acronyms

---

AI	Artificial Intelligence
AL	Active Learning
AR	Augmented Reality
BBB	Bayes By Backprop
BBN	Bayesian Neural Networks
DL	Deep Learning
DLR	German Aerospace Center
DNN	Deep Neural Network
ECE	Expected Calibration Error
ELBO	Evidence Lower Bound
GP	Gaussian Process
GPU	Graphical Processing Unit
ICP	Iterative Closest Point
IID	Independent and Identically Distributed
KFAC	Kronecker-Factored Approximate Curvature
KIT	Karlsruhe Institute of Technology
KL	Kullback-Leibler
LA	Laplace Approximation
LRA	Low Rank Approximation
LWR	Light Weight Robot
MAP	Maximum a-Posteriori
MAV	Micro Aerial Vehicle
MCMC	Markov Chain Monte Carlo
MLE	Maximum Likelihood
MLL	Marginal Log Likelihood
MLP	Multi Layer Perceptron

MND	Multivariate Normal Distribution
MoE	Mixtures of Experts
MSE	Mean Square Error
NLL	Negative Log Likelihood
NLM	Neural Linear Model
NTK	Neural Tangent Kernel
OOD	Out Of Distribution
PAC	Probably Approximately Correct
PCA	Principal Component Analysis
PNN	Progressive Neural Network
RMSE	Root Mean Square Error
UAV	Unmanned Aerial Vehicle
VI	Variational Inference
VR	Virtual Reality
XR	Extended Reality

---

## List of Symbols

---

When applicable, we denote all scalar quantities by plain letters. Matrices and vectors are printed in bold. Subscripts and superscripts are used to denote specific quantities. This list contains only the symbols which appear routinely and are therefore of importance.

$\mathcal{D}$	Data.
$\mathbf{x}$	Input data.
$\mathbf{y}$	Output data.
$\mathbf{x}^*$	New input data.
$\mathbf{y}^*$	Predictions.
$\mathbf{a}$	Activation.
$\mathbf{s}$	Pre-activation.
$\boldsymbol{\theta}$	Random variable of interest (incl. model parameters).
$\hat{\boldsymbol{\theta}}$	MAP estimate.
$f_{\boldsymbol{\theta}}$	Neural network.
$\boldsymbol{\mu}$	Mean vector of a Gaussian distribution.
$\boldsymbol{\Sigma}$	Covariance matrix of a Gaussian distribution.
$\boldsymbol{\eta}$	Information vector of a Gaussian distribution.
$\mathbf{F}$	Information matrix of a Gaussian distribution.
$\mathbf{H}$	Hessian matrix.
$\mathbf{W}$	Weight matrix.
$\mathbf{J}$	Jacobian matrix.
$\mathbf{I}$	Identity matrix.
$\mathbf{A} \otimes \mathbf{G}$	Kronecker product of two matrices.
$\mathbf{V}$	Eigenvector.
$\boldsymbol{\Lambda}$	Matrix of eigenvalues.
$\mathbf{K}$	Kernel matrix.
$\alpha, \beta, \tau, \gamma, \delta$	Any constant parameters.

$\mathcal{A}$	Acquisition function.
$\mathcal{U}$	Utility function.
$P$ & $Q$	Point clouds.
$I$	Camera Images.
$R$	Rotation matrix.
$t$	Translation vector.
$T$	Transformation matrix.
$x, y$	Arbitrary random variable.
$s$	Robot state.
$v$	Robot velocity.
$a$	Robot action.
$\tau$	Robot torques.
$F$	Wrench forces.



# Part I.

## Prologue

“Uncertainty is the necessary companion of all explorers” – Marilyn Ferguson.



# CHAPTER 1

---

## Introduction

---

Let's try to teach a child to recognize different animals, like a cat or a dog. At first, the child may struggle to tell the difference. But over time, we show them more pictures of cats and dogs. Then the child starts to notice patterns, like cats often have pointy ears, and dogs usually vary in size and may have floppy ears. Cats have retractable claws that they use for climbing and hunting, while most dogs have non-retractable claws. After seeing enough pictures, the child gets really good at telling the difference. Roughly speaking, deep learning is similar, but instead of a child, it is a computer with artificial intelligence (AI). We feed it lots of pictures, information, or examples (just like the pictures of animals), and the computer learns to spot patterns and make predictions. Deep learning is a method that relies on large amounts of data to help a computer recognize patterns, make predictions and decisions, or even understand things like pictures and speech. It's like teaching the computer to think and recognize things by looking at lots of examples, just like we often do.

As of 2025, deep learning is becoming more and more the de-facto standard for AI research and industrial practices. Ample evidence of its preponderance and popularity exists. For example, NVIDIA is a company that builds computer chips called graphical processing units (GPUs), which serve as computing infrastructure for deep learning. In 2024, NVIDIA's valuation in terms of market capitalization reached 3.566 trillion USD. This was roughly ten times more than all car manufacturers in Germany combined, namely BMW, Volkswagen, and Mercedes Benz. The Nobel Prize for Physics went to Geoffrey Hinton and John J. Hopfield for their contributions to artificial neural networks – a computational model used in deep learning ([Hopfield, 1982](#); [Rumelhart et al., 1986](#); [LeCun et al., 2015](#)). In the same year, the Nobel Prize for Chemistry was awarded to the team that developed a deep learning model called AlphaFold ([Jumper et al., 2021](#)). AlphaFold predicts protein structures and has the potential to accelerate drug development, speeding up research on cures for cancer. Natural language processing ([Vaswani et al., 2017](#)), computer vision ([Krizhevsky et al., 2012](#)), medicine ([Ardila et al., 2019](#)), natural sciences ([Wang et al., 2023a](#)), mathematics ([Trinh et al., 2024](#)), and sustainability ([Kurth et al., 2023](#)) are other exemplary areas where deep learning is leading an exciting surge of progress.

Despite the encouraging progress, however, deep learning still has several critical shortcomings. One of the shortcomings is its general inability to provide uncertainty estimates ([Gawlikowski et al., 2023](#)), thereby leading to a lack of interpretability and trustworthiness.

What is  $84322 \times 33913$ ? How confident are you in your answer?  
Any probabilities?

Let's calculate:

$$84322 \times 33913 = 2,860,151,986$$

I'm **100% confident** in this answer. It's a straightforward multiplication with no ambiguity. If you're asking in terms of probabilities:

- The probability that this result is correct is effectively **1 (or 100%)**, given that it was computed using deterministic arithmetic.

If you want, I can also show the step-by-step multiplication or verify it using another method.



**Figure 1.1.:** An example conversation with the OpenAI's chatGPT. The correct answer to the posed question should be 2859611986. However, chatGPT answers wrongly with high confidence. This conversation provides an example that standard deep learning may not be aware of its own uncertainty. Such shortcoming and its black-box nature may lead to mistrust in such models and impede its practicality in safety critical applications. A screenshot taken on the 9th July 2024.

To illustrate the problem, in figure 1.1, we provide an example from a conversation with a large language model called chatGPT, which is a well-known chat-bot based on deep learning. Here, what do we mean by the uncertainty estimates? For us, it's about the idea of using probability distributions to represent information instead of relying on one single deterministic, most likely outcome ("point estimates"). For example, when the goal is to predict a continuous variable, like in the case of regression problems, we would like to obtain a valid probability distribution such as a Gaussian with a mean and a covariance matrix. When the goal is to predict a discrete variable, like in the case of classification problems, we would like to obtain confidence scores that reflect the true probabilities. Uncertainty quantification is a set of methods that enables us to know when to trust the prediction from a model and when not. There is a famous saying, "all models are wrong, but some are useful". In this thesis, we take a stance that one type of useful model is the model with a notion of its own uncertainty, or the model that knows when it doesn't know. This problem of estimating uncertainty in deep learning is of primary focus in this thesis.

The relevance of the uncertainty quantification problem in deep learning can be further stressed. First and foremost, employing deep learning in safety-critical applications requires reliable uncertainty estimates. Typical examples include applications of deep learning models for decision-making in healthcare (Huang et al., 2024), or the autonomous driving (Feng et al., 2022a) domain. Imagine an automated cancer detection using MRI scans missing the diagnosis of cancer, or a driving assistant system failing to distinguish a pedestrian. The consequences can be catastrophic. Second, uncertainty awareness helps users comprehend and trust the decisions from deep learning. For example, Wang et al. (2023a) argues that while deep learning can accelerate scientific discoveries, measuring the

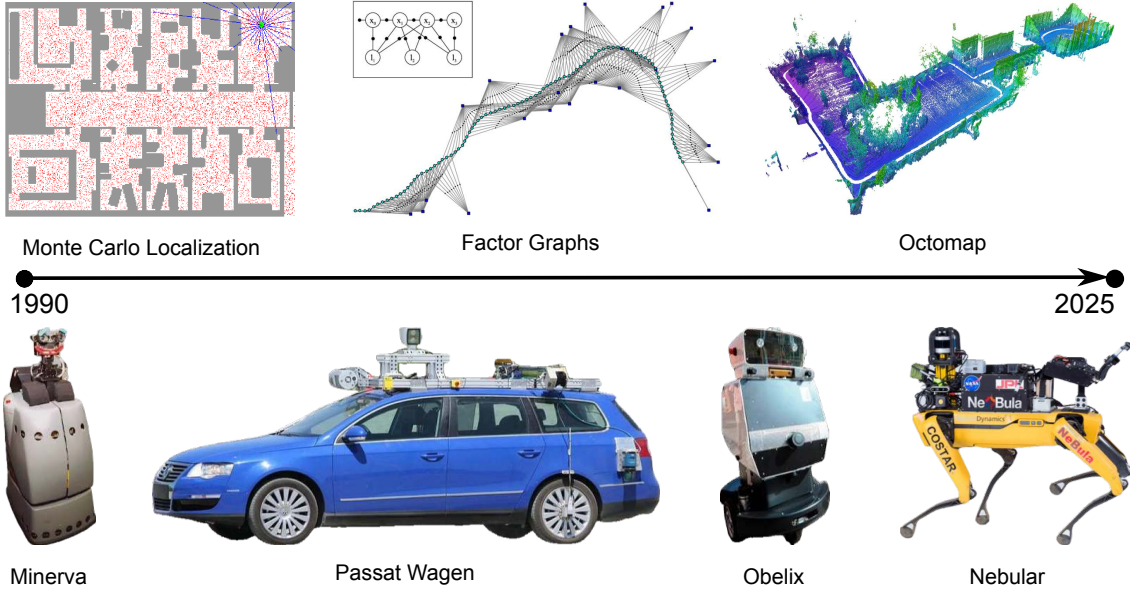
levels of uncertainty is essential for ensuring that we do not rely heavily on potentially flawed results. Lastly, uncertainty estimates can also be applied to improve the machine learning algorithms in deep learning. Few examples are in the problem of (a) active learning where an algorithm chooses the most informative data points to learn from (Settles, 2012; Ren et al., 2021), (b) Bayesian optimization where an optimizer efficiently finds the solution of a complex objective function (Frazier, 2018; Wang et al., 2023b), (c) bandits and explorations in reinforcement learning where an agent must balance exploration (trying new actions to gather more information) and exploitation (choosing actions that maximize known utility) (Sutton, 2018). What is common to these use cases is decision-making under uncertainty.

Among others, robotics is of particular interest in this thesis. To integrate deep learning into robotics, the aforementioned traits such as safety, trustworthiness, and decision-making under uncertainty are highly desirable. This is because robots are physical machines and their actions have consequences in the real world. To this end, the general role of uncertainty in robotics has been historically emphasized, leading to an area called probabilistic robotics. To address the generic problem of quantifying uncertainty in deep learning, this thesis shows that the field of probabilistic robotics can provide new perspectives for further progress.

## 1.1. Probabilistic Robotics

Probabilistic robotics (Thrun et al., 2002) is a subfield of robotics that deals with various uncertainties the robot inevitably encounters in the physical world. To scale robotic systems beyond assembly lines in factories, its key idea is to build the robots that pay tribute to the uncertainties when connecting the robot’s perception and action. That is, robot perception is formulated as a probabilistic reasoning problem, in which the robot has to represent its uncertainty when estimating a quantity of interest through sensor measurements. Robot control is a decision-making problem in which the robot has to maximize its utility in the face of uncertainty. Here, control actions can be the robot’s behavior generation towards safety or even the reduction of uncertainty by online adaptation of the perception system. In this way, probabilistic robotics provides a unifying perspective for the integration of perception, learning, and control within one physical robot, which can contend with complex environments and novel tasks. Thus, probabilistic robotics enables us to use less prior knowledge about unstructured environments and less constrained specification of task execution. The major hypothesis is that the robot with its own notion of uncertainty will scale better than the robot that represents information without any notion of uncertainty.

In the past, several researchers developed probabilistic methods for robotics. One of the early attempts was on robotic manipulation in assembly lines (Brooks, 1982). Soon after, Cheeseman et al. (1987) proposed to represent the world model of object relations using probability theory. This idea of embedding uncertainty into the robot’s representation of the world was utilized by well-known techniques such as probabilistic grid maps (Elfes, 1989; Burgard et al., 1996), elevation maps (Kweon et al., 1989), multi-level surface maps (Triebel et al., 2006), and OctoMaps (Hornung et al., 2013). Given the map of the environments, probabilistic methods for state estimation (Fox et al., 1999; Dellaert et al., 1999) and planning (Roy et al., 1999; Kaelbling et al., 1998) were also investigated. In the 2000s, probabilistic methods were actively applied to the problem of Simultaneous Localization and Mapping (SLAM). SLAM involves tracking the robot’s location within unknown environments while building a map of that environment (Cadena et al., 2016). Some examples were FastSLAM (Montemerlo et al., 2002), SEIF (Thrun et al., 2004), and graph SLAM (Grisetti et al., 2010). Notably, Dellaert & Kaess (2006) observed that the



**Figure 1.2.:** Past achievements in the research area of uncertainty in robotics. Probabilistic robotics entails the history of developing methods that are well suited for their applications to robotics. Using these methods, several robotic systems were built to demonstrate new applications.

information matrix, i.e., the inverse of the covariance matrix, can be factorized into sparse matrices. Such sparse matrices could be incrementally updated (Kaess et al., 2008), or represented as a graphical model called the Bayes tree that can solve the nonlinear least squares problem incrementally (Kaess et al., 2012). These developments led to the GT-SAM library (Dellaert, 2012) – one of the hallmarks in probabilistic robotics for its wide adoption in practice. Even today, probabilistic methods are still investigated for perception (Sodhi et al., 2022), planning (Curtis et al., 2024; Ha et al., 2020), control (Berkenkamp et al., 2017), simulation (Ramos et al., 2019) and many more topics in robotics research.

Along with the progress in probabilistic methods, we can see evaluation of these methodologies at the system level. One of the prominent examples is Rhino (Buhmann et al., 1995; Burgard et al., 1999) and Minerva (Thrun et al., 2000). Particularly, in the fall of 1998, an interactive tour guide robot, Minerva, was successfully deployed in the Smithsonian Museum. During its two weeks of operation, the robot traversed over 44 km, interacting with thousands of visitors. Thrun et al. (2000) argues that the key technical enabler was Minerva’s probabilistic algorithms. From perception to action, Minerva relied on probabilistic methods such as Monte-Carlo localization (Fox et al., 1999), occupancy grid maps (Elfes, 1989; Burgard et al., 1996), and planning under uncertainty (Roy et al., 1999; Kaelbling et al., 1998). In this way, Minerva showed how probabilistic approaches can enable robust operations in complex environments, where approaches that ignored a robot’s uncertainty could not scale back then. Another example is from the domain of autonomous driving. In Freiburg, researchers demonstrated autonomous parking in a complex real parking garage (Kummerle et al., 2009). The self-driving car utilized a probabilistic localization technique (Kummerle et al., 2008) on a mapping framework of Triebel et al. (2006). A robot called Obelix is another probabilistic robot, which traversed over 3 km in the city of Freiburg autonomously, despite the human presence in the streets (Kummerle et al., 2013). One of the most recent systems is from the DARPA Subterranean Challenge, where multiple robots had to navigate and map complex underground environments, such as tunnels, caves,

and urban infrastructure. Notably, the JPL’s NeBula (Morrell et al., 2022) devised an uncertainty-aware autonomy solution for the perception and planning of robots.

New technologies bring not only opportunities, but also new problems. Traditionally, probabilistic robotics focused on uncertainty due to imperfect sensor measurements. Since robotic perception relied on direct processing of noisy sensor measurements, modeling and propagation of the sensor noise was crucial back then (Thrun et al., 2002). However, due to the popularity of deep learning (Goodfellow et al., 2016), neural networks are nowadays being used more and more for perception tasks. This trend of AI-based robotics is urging for paradigm shifts in probabilistic robotics. That is, instead of reasoning about uncertainty in sensor measurements, new tools are needed to compute uncertainties due to imperfections in deep learning models. Similar to how probabilistic robotics developed in a classical realm, these new tools need to be validated and refined further based on their real-world applications. In this thesis, we approach the problem of uncertainty quantification in deep learning, emphasizing this balanced interplay between methods and systems.

## 1.2. Problem Statement

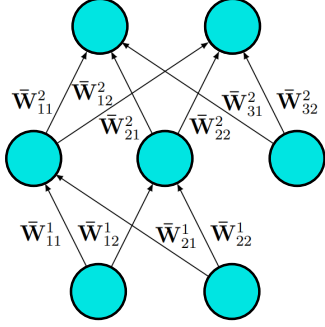
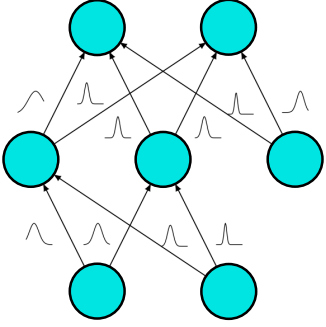
The main objectives of this thesis are twofold. On the fundamental side, we aim to develop probabilistic methods to quantify uncertainty in deep learning. A concrete goal is to advance the state-of-the-art methods in the metrics relevant to robotics while providing theoretical foundations behind the proposed methodologies. At the system level, we aim to ground these methods in concrete applications of robotics and demonstrate possibilities for new applications. For probabilistic robotics, the goal is to demonstrate that learning systems with their notion of uncertainty can scale better than systems that ignore this uncertainty. With these results, this thesis also aims at consolidating the already probabilistic nature of many robotics applications with novel probabilistic representations of deep learning.

For this, we build on Bayesian statistics. To explain, imagine our goal is to predict if it’s going to rain tomorrow. Before anything, we might say, “well, in winter, it rains about 55% of the time here”. This is our *prior belief*. Then, we check the weather forecast. The forecast says “dark clouds are forming” which gives us *new evidence or data*. Now, we can make predictions by updating our prior belief with this new evidence. Indeed, we may predict: “there is a high chance that it rains tomorrow”. This example illustrates the intuition behind Bayesian statistics, which uses Bayes’ Theorem to update one’s belief about an event as new evidence arrives. More formally, utilizing one’s belief about an event, we can make predictions  $\mathbf{y}^*$  for new incoming data  $\mathbf{x}^*$ . Given a random variable  $\boldsymbol{\theta}$  and data  $\mathcal{D}$ :

1. We define our prior belief about the event  $\boldsymbol{\theta}$ :  $p(\boldsymbol{\theta})$ .
2. Once new evidence arrives, we update our belief:  $p(\boldsymbol{\theta}|\mathcal{D}) = \frac{p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathcal{D})}$ .
3. Predict for the new datum  $(\mathbf{x}^*, \mathbf{y}^*)$ :  $p(\mathbf{y}^*|\mathbf{x}^*, \mathcal{D}) = \int p(\mathbf{y}^*|\mathbf{x}^*, \boldsymbol{\theta})p(\boldsymbol{\theta}|\mathcal{D})d\boldsymbol{\theta}$ .

The first step is called prior specification, while the second step is called posterior inference. With Bayes’ theorem, we obtain a posterior distribution  $p(\boldsymbol{\theta}|\mathcal{D})$  which is a function of the likelihood  $p(\mathcal{D}|\boldsymbol{\theta})$ , the prior  $p(\boldsymbol{\theta})$ , and the evidence  $p(\mathcal{D})$ . The last step is called predictions through marginalization. The marginalization involves averaging over all the possibilities, weighted by the posterior probabilities. The resulting distribution  $p(\mathbf{y}^* | \mathbf{x}^*, \mathcal{D})$  captures uncertainty in predictions  $\mathbf{y}^*$ . The same procedure can be applied to deep learning, hence called “Bayesian deep learning”. As in table 1.1, the model parameters in deep learning are treated as random variables. This is plausible because our model itself is uncertain.



Standard Deep Learning	Bayesian Deep Learning
	
<ol style="list-style-type: none"> <li>1. User defines an architecture <math>f_{\theta}</math>.</li> <li>2. Training results in <math>\hat{\theta} \in \operatorname{argmax} p(\theta \mathcal{D})</math>.</li> <li>3. Predictions result in <math>\mathbf{y}^* = f_{\theta}(\mathbf{x}^*)</math>.</li> </ol>	<ol style="list-style-type: none"> <li>1. User defines a prior <math>p(\theta)</math> for <math>f_{\theta}</math>.</li> <li>2. Training results in posterior <math>p(\theta \mathcal{D})</math>.</li> <li>3. Predictions result in <math>p(\mathbf{y}^* \mathbf{x}^*, \mathcal{D})</math>.</li> </ol>

**Table 1.1.:** Differences between standard deep learning and Bayesian methods are highlighted. In deep learning, the user defines the structure of learning process through a neural network architecture. Then, given training data  $\mathcal{D}$ , the outcome of training is a single set of model parameters  $\hat{\theta}$ . Predictions are inherently deterministic. In contrast, Bayesian methods define priors over the model parameters, in addition to the model structure. The outcome of training is a model that is a probability distribution. Predictions contain uncertainty estimates. The illustrations show fully-connected layers. However, the methodology applies to other architectures as well.

Applying Bayesian principles to deep learning provides several advantages. First, domain knowledge can be incorporated into the learning algorithm through priors (Fortuin, 2022). Therefore, choosing appropriate priors allows knowledge transfer across diverse domains and tasks for generalization and uncertainty estimation (Finn et al., 2018; Wang et al., 2024). Second, Bayesian methods can quantify uncertainty over the deep learning models. “Knowing what our models don’t know” helps us capture situations when test conditions are underrepresented in the training data. This phenomenon of distributional shifts routinely occurs when a robot tries to contend with complex environments and novel tasks (Sünderhauf et al., 2018). Third, for predictions, Bayesian methods average different hypotheses instead of relying on point estimates. Predictions through marginalization, also known as Bayesian Model Averaging, can improve accuracy and uncertainty estimates while avoiding overfitting (Wilson & Izmailov, 2020). Lastly, deep learning can be combined with existing tools from Bayesian machine learning. For example, deep learning can benefit from the Bayesian way of model compression (Louizos et al., 2017), model selection (Immer et al., 2021a), active learning (Settles, 2012), reinforcement learning (Sutton, 2018) and many more.

However, the applications of Bayesian statistics to deep learning are challenging. First, neural networks are black box models where their parameters are uninterpretable and lie in high-dimensional space. As a result, prior specification to encode domain knowledge is non-trivial. Thus, uninformative priors are often used despite the signs of prior misspecification (Fortuin, 2022). Second, modern deep learning relies on large amounts of data and over-parameterized architectures with billions of parameters. Consequently, inference of complex posteriors is computationally intractable. To this end, simplified representations such as Bernoulli distributions and mean field approximations are commonly used, inducing crude approximations of the true posteriors (Gawlikowski et al., 2023). Lastly, predictions through marginalization involve averaging over multiple hypotheses for a single test input.



Each hypothesis is typically obtained by drawing a sample from the posterior and running a forward pass through the deep learning model. Such operations are computationally expensive, especially for robots with constrained resources (Gawlikowski et al., 2023).

In consequence, this thesis is structured around three key research questions:

1. How to specify more meaningful priors for uncertainty and generalization?
2. How to infer complex posteriors while scaling to large dataset and models?
3. How to efficiently predict through marginalization by avoiding any sampling?

Addressing these research questions can bring several promises of Bayesian methods from a theoretic construct to practical implementations. Together, these research questions also point towards a single problem: how to obtain reliable uncertainty estimates from deep learning models, which are suitable for their applications in robotics and beyond?

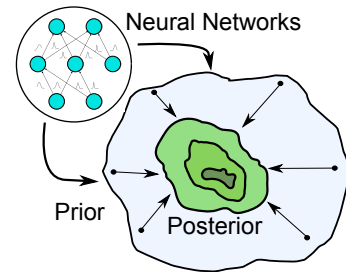
### 1.3. Contributions

To recap, this thesis addresses the problem of uncertainty in deep learning. To do so, our objective is to advance the current state-of-the-art in a relevant way for robotics and further provide system-level demonstrations that showcase the effectiveness of probabilistic methodologies in real-world applications. Applying Bayesian principles to deep learning is potentially a promising way to reason about uncertainty. Yet, several challenges exist for each sub-problem of Bayesian statistics, namely (a) prior specification, (b) posterior inference, and (c) predictions through marginalization. In this thesis, we therefore propose new methods for each aforementioned challenge. Specifically, we devise Bayesian methods that enhance generalization through priors, scale inference to complex posteriors, and accelerate marginalization for efficient predictions. Moreover, we show practical implementations of these methodologies. From humanoid robots recognizing deformable objects, aerodynamic analysis of stratospheric and manned helicopter flights, and semi-autonomous aerial manipulation in challenging environments, to shared autonomy using haptics and extended reality, we contribute advances across multiple systems and applications.

Below, we expand and explain these contributions in more detail.

#### 1.3.1. Main Contributions

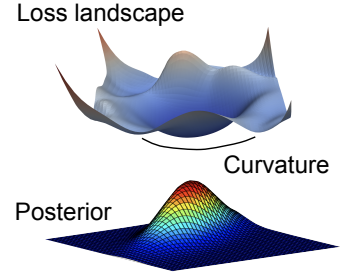
**On priors.** A method for prior specification is presented. Specifically, we propose a novel prior learning method to enhance generalization and uncertainty estimation in Deep Neural Networks (DNNs). The central concept is to leverage the scalable and structured posteriors of neural networks as informative priors with generalization guarantees. Our learned priors offer expressive probabilistic representations at a large scale, akin to the Bayesian equivalents of pre-trained models on ImageNet, while also delivering meaningful generalization bounds. We extend this approach to a continual learning framework, where the beneficial properties of our priors are particularly useful. Our key technical enablers are: (1) sums-of-Kronecker-product computations, and (2) the derivation and optimization of tractable objectives that improve generalization bounds. Empirically, we demonstrate the effectiveness of this method for uncertainty estimation and generalization.



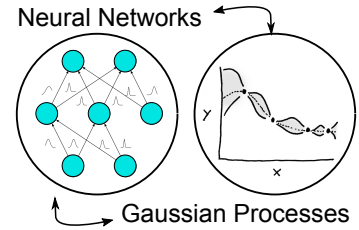
**Figure 1.3.:** Learning priors.

**On posteriors.** A method for posterior inference is presented. Concretely, we propose a sparse representation of model uncertainty for DNNs, where the parameter posterior is approximated using an inverse formulation of the Multivariate Normal Distribution (MND), also known as the information form. The core insight of our work is that the information matrix, which is the inverse of the covariance matrix, typically exhibits sparsity in its spectrum. As a result, dimensionality reduction techniques, such as low-rank approximations (LRA), can be effectively applied. To achieve this, we develop a novel sparsification algorithm and derive an efficient analytical sampler. Consequently, we demonstrate that the information form can be applied at scale to represent model uncertainty in DNNs. Among many experiments, we show that our method scales to large architectures on ImageNet, which contains more than 14 million labeled images.

**On predictions.** A method for efficient marginalization is presented. Our method features reliable uncertainty estimates based on a mixture of Gaussian Process (GP) experts (or local GPs), which are combined with a DNN, so that the resulting predictor is also accurate. We theoretically prove that neural networks can be cast as a mixture of GP experts with a neural network-based kernel called the Neural Tangent Kernel (NTK). We further show how this theory can be used to obtain predictive uncertainties fast and reliably. To do so, a divide-and-conquer strategy and an efficient predictive algorithm are devised. At test time, our predictive model is efficient and, by design, maintains the predictive power of DNNs. Empirically, we not only show advancements over the state-of-the-art, but also demonstrate its applicability to a micro aerial vehicle with limited onboard computations.



**Figure 1.4.:** Information in the curvature of the loss.

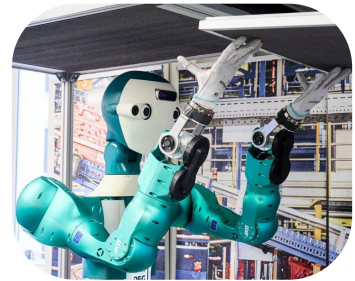


**Figure 1.5.:** DNN & GP

### 1.3.2. System-level Evaluation and Extensions

Now, we extend the developed methods in practice. We show that these methods improve robustness of deep learning on real robots (system 1 and 4). Uncertainty estimates can also address the lack of annotated training data in real world applications (system 2 and 3).

**System 1.** This work extends the proposed prior learning method. For that, we develop CLEVER, a stream-based active learner for robust semantic perception using DNNs. This system seeks human support when encountering failures and adapts the model online based on human instructions. In this way, CLEVER can eventually complete the given semantic perception tasks. The key enabler is our Bayesian formulation that encodes domain knowledge through priors. Empirically, through a user validation study with 13 participants, we first demonstrate that our system can perform semantic perception on various objects. Second, we showcase CLEVER’s capabilities on a humanoid robot for the semantic perception of deformable objects. To the best of our knowledge, this is the first work to implement active learning from streams of data on a real robot, providing



**Figure 1.6.:** Humanoid.

evidence that the robustness of DNN-based semantic perception improves in practice.

**System 2.** We extend our posterior inference method to tackle a real-world problem in experimental aerodynamics. Concretely, tufts, strips of wire or rope, are frequently used to visualize the local flow direction around aerial systems. We propose a computer vision system that automatically extracts shapes of tufts from images taken from an unmanned aerial vehicle (UAV) and manned helicopter flights. A semantic segmentation pipeline is devised consisting of three uncertainty-driven modules: (a) active learning for object detection, (b) image matching for object classification, and (c) weakly supervised instance segmentation. This probabilistic approach facilitate a learning process without any costly annotations of semantic segmentation masks, unlike the end-to-end supervised deep learning. We substantiate our design decisions by showing how improvements in uncertainty estimation enable more efficient use of training data. Real-world demonstrations of the devised system is further provided.



**Figure 1.7.:** Understanding manned helicopter flight.

**System 3.** We extend our posterior inference method to develop a telepresence system for aerial manipulation. The system features a haptic device and a Virtual Reality (VR) interface that provides real-time 3D views of the robot’s workspace. This system is based on our perception pipeline, which includes a pool-based active learning system that improves sample efficiency of DNN-based object detection. Experimental results demonstrate how estimating algorithmic failures and uncertainty helps address our scenarios. In particular, with data collected from a long-term deployment of the robot, we show that our posterior inference method improves the active learning performance when compared to a popular alternative of [Gal & Ghahramani \(2016\)](#). Furthermore, field experiments, including night operations and involving over 70 successful tasks with the DLR cable-Suspended Aerial Manipulator (SAM), demonstrate the system’s effectiveness and viability of our concept for future industrial applications.



**Figure 1.8.:** Aerial manipulation at night.

**System 4.** The GP-based uncertainty estimation technique is extended here. In particular, we develop SPIRIT – a shared autonomy system for robust aerial manipulation under uncertainty. A novel feature of SPIRIT is its probabilistic approach, allowing users to interact with a remote robot through an explicit representation of uncertainty in deep learning. Specifically, from perception modules that pervasively rely on DNNs nowadays, we obtain the uncertainty estimates to decide the level of autonomy, e.g., the users can switch from semi-autonomous manipulation to pure teleoperation if the predictions are highly uncertain. This way, we integrate uninterpretable DNN methods more safely while leveraging their state-of-the-art performance in the real world. A technical enabler is our partitioned probabilistic approach to point cloud registration with the NTK. Empirically, an ample evidence is provided to show how



**Figure 1.9.:** Shared autonomy for aerial manipulation.

embracing uncertainty in deep learning results in robust real-world operations.

In summary, the major contributions of this thesis are as follows.

**On priors:**

1. We propose to exploit scalable and structured posteriors of neural networks as informative priors with generalization guarantees.
2. We derive the sums-of-Kronecker-product computations that provably converge to the rank-one optimality. PAC-Bayes objectives are derived for a tractable generalization.
3. We develop a stream-based active learner for robust semantic perception. Using deep learning, the active learning is demonstrated on a physical robot for the first time.

**On posteriors:**

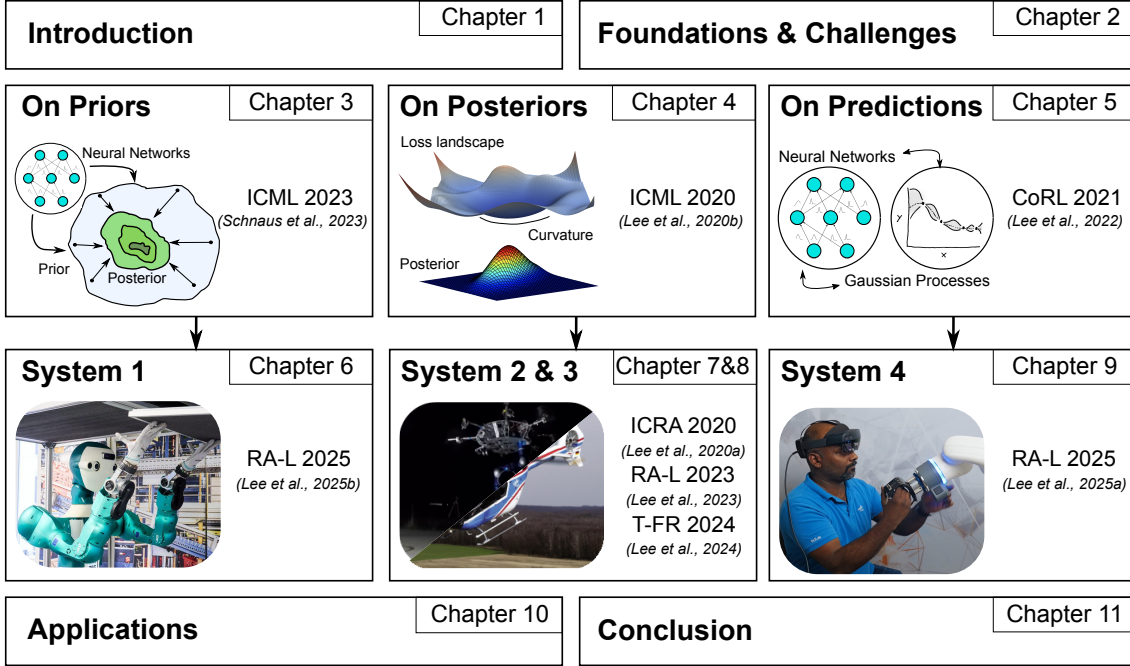
4. We propose to represent the posterior of neural networks in sparse information form, as a scalable technique to perform probabilistic inference.
5. We derive the low-rank sampling computations that allow a direct sampling from our sparse information formulation: Kronecker eigen-decomposition plus diagonal form.
6. We develop a computer vision system that enables in-flight aerodynamic analysis of a helicopter and stratospheric drone using the probabilistic method.
7. We develop a perception system for aerial manipulation in challenging environments. State-of-the-art aerial manipulation is achieved using the probabilistic method.

**On predictions:**

8. We propose to efficiently estimate predictive uncertainty of neural networks with a sparse variant of GPs called mixture of GP experts, also known as local GPs.
9. We derive a theoretic connection between DNNs and mixture of GP experts with the NTK, adding a new result to the universal approximation theory of DNNs.
10. We develop SPIRIT, a shared autonomy for aerial manipulation under uncertainty. The first version of this system won the Finalist of KUKA Innovation Award 2023.

With the aforementioned contributions, this thesis provides both the methodological advances to quantify uncertainty in deep learning, as well as systems' contributions to show that uncertainty awareness can increase the scalability of robots in practice. By addressing several challenges, we demonstrate that (a) meaningful priors can be specified to incorporate domain knowledge, (b) complex posteriors can be scalably obtained for large datasets and models, and (c) marginalization can be made run-time efficient by avoiding any sampling from the posterior. These contributions bridge the probabilistic robotics with recent advances in artificial intelligence, paving the way towards safe and trustworthy robotic systems that are capable of intelligent decision-making under uncertainty.

Furthermore, the developed methods and systems are being used in the national and international projects of the German Aerospace Center (DLR). We successfully transferred our contributions to the domain of future manufacturing, planetary exploration, and service robotics with humanoids. Multiple perception tasks such as SLAM, object detection, and localization of linear deformable objects are also showcased. We also outline our efforts in



**Figure 1.10.:** Structure of this thesis and interconnections of each chapters.

developing a software infrastructure as a final result of this thesis. Many of these cover experimentation outside the lab. Therefore, we show that our contributions do not overfit to specific scenarios, but they can be used to address multiple use cases more broadly.

## 1.4. Outline

Figure 1.10 shows the outline of the thesis, the interrelation of each chapter, and how the respective chapters are related to the aforementioned contributions. Concretely, chapter 2 covers background information on relevant topics such as uncertainty in robotics, Bayesian statistics, challenges of Bayesian Deep Learning, and current practices in benchmarking uncertainty quantification methods. This chapter also provides a brief review of the existing works, which are classified into priors, posteriors, and predictions. Chapters 3, 4 and 5 present our methodological contributions on priors, posteriors, and predictions, respectively. Chapters 7, 8, 6 and 9 provide our system contributions that utilize the developed methods. Then, chapter 10 disseminates the contributed methods and systems in other use cases. Lastly, chapter 11 concludes the thesis. The research presented in this thesis addresses one single problem: uncertainty in deep learning. As several methodologies and systems have been built around one problem, we compare and report the lessons learned. Despite the progress in understanding the given problem, many limitations remain. We thereby discuss opportunities for extensions and exciting venues for further research.

The research findings reported in this dissertation resulted in eleven publications (eight lead author papers and three co-authored). The list of these publications can be found in table 1.2. The lead author publications form the main body of each chapter. The references are indicated in both table 1.2 and figure 1.10. Furthermore, in total, eleven more publications have been co-authored (two lead author papers and ten co-authored), which are related to the topic of this thesis broadly, but are not incorporated into the thesis.



A complete list of publications is provided in appendix D at the end of the thesis.

Reference	Details
<a href="#">Lee et al. (2025a)</a>	<b>Jongseok Lee</b> , Ribin Balachandran, Harsimran Singh, Jianxiang Feng, Hrishik Mishra, Marco De Stefano, Rudolph Triebel, Alin Albu-Schaeffer and Konstantin Kondak, SPIRIT: Shared Autonomy for Robust Aerial Manipulation under Uncertainty in Deep Learning. Submitted to IEEE Robotics and Automation Letter (RA-L).
<a href="#">Lee et al. (2025b)</a>	<b>Jongseok Lee</b> , Timo Birr, Rudolph Triebel and Tamim Asfour, Stream-based Active Learning for Robust Semantic Perception from Human Instructions. IEEE Robotics and Automation Letter (RA-L), 2025.
<a href="#">Lee et al. (2024)</a>	<b>Jongseok Lee</b> , Ribin Balachandran, Konstantin Kondak, Andre Coelho, Marco De Stefano, Matthias Humt, Jianxiang Feng, Tamim Asfour and Rudolph Triebel, Introspective Perception for Long-term Aerial Telemanipulation with Virtual Reality, IEEE Transactions on Field Robotics (T-FR), 2024.
<a href="#">Schnaus et al. (2023)</a>	Dominik Schnaus*, <b>Jongseok Lee*</b> , Daniel Cremers and Rudolph Triebel, Learning Expressive Priors for Generalization and Uncertainty Estimation in Neural Networks, In Proc. of the International Conference on Machine Learning (ICML), 2023.
<a href="#">Lee et al. (2023)</a>	<b>Jongseok Lee*</b> , Jurrien Olsman* and Rudolph Triebel, Learning Fluid Flow Visualizations from In-flight Images with Tufts, IEEE Robotics and Automation Letter (RA-L), 2023.
<a href="#">Lee et al. (2021)</a>	<b>Jongseok Lee</b> , Jianxiang Feng, Matthias Humt, Marcus G. Muller and Rudolph Triebel, Trust Your Robots! Predictive Uncertainty Estimation of Neural Networks with Sparse Gaussian Processes, Conference on Robot Learning (CoRL), 2021.
<a href="#">Lee et al. (2020b)</a>	<b>Jongseok Lee</b> , Matthias Humt, Jianxiang Feng and Rudolph Triebel, Estimating Model Uncertainty of Neural Networks in Sparse Information Form, In Proc. of the International Conference on Machine Learning (ICML), 2020.
<a href="#">Lee et al. (2020a)</a>	<b>Jongseok Lee</b> , Ribin Balachandran, Yuri S. Sarkisov, Marco De Stefano, Andre Coelho, Kashmira Shinde, Min Jun Kim, Rudolph Triebel, and Konstantin Kondak, Visual-Inertial Telepresence for Aerial Manipulation, In Proc. of the IEEE/RSJ International Conference on Robotics and Automation (ICRA), 2020.
<a href="#">Gawlikowski et al. (2023)</a>	Jakob Gawlikowski, Cedrique Rovile Njieutcheu Tassi, Mohsin Ali, <b>Jongseok Lee</b> , Matthias Humt, Jianxiang Feng, Anna Kruspe, Rudolph Triebel, Peter Jung, Ribana Roscher, Muhammad Shahzad, Wen Yang, Richard Bamler and Xiao Xiang Zhu, A Survey of Uncertainty in Deep Neural Networks, Artificial Intelligence Review, 2023.
<a href="#">Feng et al. (2022b)</a>	Jianxiang Feng, <b>Jongseok Lee</b> , Maximilian Durner and Rudolph Triebel, Bayesian Active Learning for Sim-to-Real Robotic Perception, In Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2022.
<a href="#">Feng et al. (2023)</a>	Jianxiang Feng, <b>Jongseok Lee</b> , Simon Geisler, Stephan Gunemann and Rudolph Triebel, Topology-Matching Normalizing Flows for Out-of-Distribution Detection in Robot Learning, Conference on Robot Learning (CoRL), 2023.

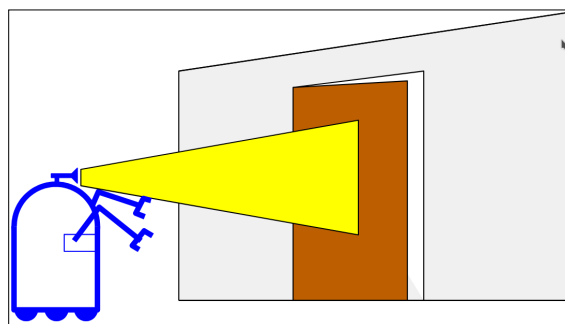
**Table 1.2.:** Main publications on the topic of this thesis. \* denotes equal contributions.

In this chapter, we introduce the problem of uncertainty quantification in deep learning within the context of robotics. Then, as we build upon Bayesian statistics, the relevant foundations on this topic are provided. To better understand the problem, we dive into the challenges in applying Bayesian statistics to deep learning. In light of these challenges, a review of the current state-of-the-art is provided. Benchmarking and measuring the quality of uncertainty estimates is a non-trivial topic since no ground truth exists. We therefore discuss current practices and standards of evaluating the quality of uncertainty estimates.

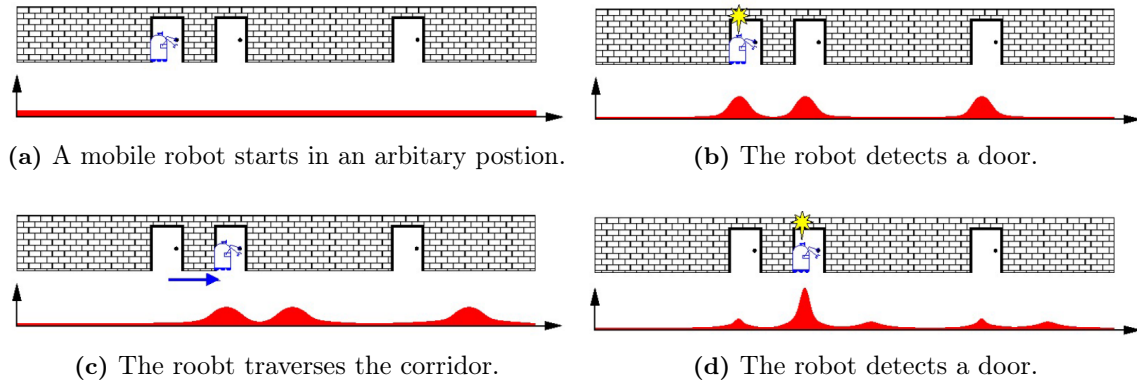
## 2.1. Uncertainty in Robotics & Deep Learning

Imagine a mobile robot. As illustrated in figure 2.1, the robot has sensors to detect doors. The robot can move using its wheels. To simplify the problem, the robot can only move in one dimension. All the wheels are equipped with position encoders. So, the robot can roughly know the distance it has traversed over time. Now, by utilizing this information, the robot is asked to localize itself within an environment. Let's say, the robot is in the corridor with three doors (see figure 2.2) and the robot needs to tell its location with a one-dimensional coordinate. The robot has a map of the corridor and how each door is placed with respect to each other. But the robot does not know its starting position.

At time zero, the robot starts next to the first door. The sensors are not activated yet.



**Figure 2.1.:** A mobile robot with sensor to detect doors ([Burgard, 2021](#)).



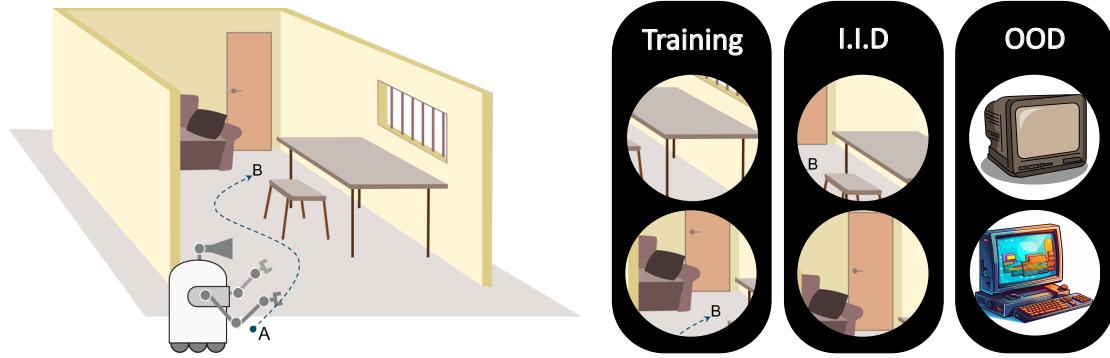
**Figure 2.2.:** A probabilistic approach to localization of a mobile robot (Burgard, 2021).

Can the robot tell its own location at this time? The answer is no. By construction, the starting position is not given to the robot. Therefore, at this time, the robot does not know its own position within the corridor. Then, how should we represent this situation mathematically? The most natural way is to use probability distributions (Thrun et al., 2002). Since the robot's position is unknown, it can be, equally likely, anywhere in the corridor. Now, we activate the sensors and the robot detects a door. From the viewpoint of the robot, the door it detected can be any of the three doors in the corridor. Again, the most natural way to represent this information is probability distributions, i.e., the robot is likely to be in front of any doors. Next, while the robot is traversing the corridor, it has detected another door, which happens to be the second one in this example (see figure 2.2). Because the robot knows the distance between each door and how much it has moved from the first door, the robot can now be confident that it's in front of the second door.

This toy problem introduces the key insights behind uncertainty and probabilistic representations for robotics. First, the robots may encounter ambiguous situations, which may arise from the problem definition itself. In the given toy problem, the robot has no way of localizing itself at the starting position and these ambiguities cannot be represented deterministically. Moreover, while we assumed that the robot can perfectly detect doors and know how much it has traversed, real-world applications are more complicated than this toy problem. Sensors are inherently noisy and limited in what they can perceive (Thrun et al., 2002). Robot actuation, especially low-cost hardware, is also unpredictable due to effects such as wear-and-tear and error of the controller (Thrun et al., 2002). These two factors can give rise to uncertainty, which cannot be ignored in robotics.

Similarly, deep learning is inherently error-prone due to various uncertainties from the operations of a robot. To explain, let's expand the previous toy problem. Now, the mobile robot is in a three-dimensional world (see figure 2.2). The robot is equipped with a camera. Instead of detecting doors, the robot recognizes various objects in a room using computer vision. For the recognition of objects, a deep learning model is trained using visual data. The data is collected by the robot, knowing the presence of particular objects, such as chairs and desks. In this scenario, when deployed in the same room, the robot can recognize these given objects with a certain level of accuracy. However, even when test conditions resemble the training, the model will likely not achieve 100% training and test set accuracy due to several factors such as overfitting, label errors, class ambiguity, imperfect optimizers, and representation power. Furthermore, imagine that the robot is in a different room that contains different objects such as computers and televisions. The model has never seen these objects before and will fail to recognize them. Also, slight changes in lighting and





**Figure 2.3.:** A modern mobile robot which navigates in a room. Now, with a camera and computer vision, the robot recognizes different objects in the room. The robot can typically encounter known objects that is independent and identically distributed (I.I.D) to the training data. A mobile robot, in a different room, may encounter distribution shifts where test conditions are not well represented in the training data. Out-of-distribution (OOD) is one example of distribution shifts.

background can cause distribution shifts, where the test conditions are underrepresented in the training data. These can reduce the model’s accuracy for the robot.

Notably, Bayesian statistics emphasizes two major sources of uncertainty ([Der Kiureghian & Ditlevsen, 2009](#); [Kendall & Gal, 2017](#)). Aleatoric uncertainty (or data uncertainty) comes from the inherent randomness or noise in the data itself. Few examples are sensor noise, ambiguous or overlapping classes, label noise, or even intrinsic randomness in the processes. This type of uncertainty is known to be irreducible even at the limits of the data. In contrast, epistemic uncertainty (or model uncertainty) arises due to the model’s lack of knowledge about the true data distribution. For example, a model is uncertain when not enough data is available or could not learn the underlying patterns well. Improving the model or collecting more training data can reduce this uncertainty. So, in robotics, when do each source of uncertainty play an important role? Aleatoric uncertainty sets a limit on achievable accuracy and is useful when large amounts of data are available ([Kendall & Gal, 2017](#)). From the example from figure 2.3, this may correspond to the first scenario, when the robot is in the first room with chairs and desks. On the other hand, out-of-distribution and distribution shift scenarios are better captured by epistemic uncertainty ([Kendall & Gal, 2017](#)). This may correspond to the second scenario, when the robot is in the second room with computers and televisions. Aleatoric uncertainty is always present and expected, but epistemic uncertainty indicates when the robot is out of its comfort zone or faced with novel situations. These situations can create unexpected failures and safety risks.

Having introduced several concepts on uncertainty in robotics and deep learning, we now closely examine Bayesian statistics – a key theory commonly used throughout this thesis.

## 2.2. Bayesian Statistics

What is probability? In traditional statistics, the probability is interpreted as the frequency of an event occurring over many independent trials (hence, also called Frequentist statistics):

$$\text{Probability} = \lim_{N \rightarrow \infty} \frac{\text{Number of events happening in } N \text{ trials}}{N}.$$

This interpretation seems plausible. But, what if we are interested in an event, of which we cannot conduct multiple independent trials? [Krause & Hübotter \(2025\)](#) gives an example

	Frequentist Statistics	Bayesian Statistics
<b>Definition of Probability</b>	Probability is the long-term frequency of an event. Probability is objective.	Probability is our belief about the likelihood of an event. Probability is subjective
<b>Role of Data &amp; Parameters</b>	Data is random and varies. The model's parameters are deterministic.	Once observed (or sampled), data is fixed. The model's parameters are random variables.
<b>Prior Information</b>	Analysis is based on the data at hand (and likelihood). No prior information is used.	Analysis starts with a prior distribution, which is updated once data is observed.
<b>Confidence Interval Vs Uncertainty</b>	A 95% confidence interval implies that if the experiment were repeated numerous times, about 95% of the resulting intervals would capture the true parameter value. Likewise, p-values help determine whether the observed data aligns with what would be expected under the null hypothesis.	The posterior distribution fully represents the uncertainty surrounding the parameter of interest. For instance, a 95% credible interval indicates that, based on the observed data and prior knowledge, there is a 95% chance that the true parameter falls within that range.

**Table 2.1.:** Few key differences between Frequentist statistics and Bayesian statistics.

outcome: “Person X will live for at least 80 years”. Indeed, in this case, there is no way of conducting multiple independent trials. In Bayesian statistics, unlike the Frequentist,

probability is a subjective analysis of the likelihood that an event might happen.

It’s our *belief* and hence, probability does not exist outside the mind (De Finetti, 1970). This interpretation of probability as a measure of our belief or uncertainty allows us to analyze events, where independent experiments are impossible or subjective in nature.

Several differences between Frequentist and Bayesian statistics should be noted further. Table 2.1 provides a summary. One important conceptual difference is the role of data and the model. In Frequentist statistics, by definition, the model (or its parameters) is unknown, but fixed. It’s data that varies – a true distribution is a black box and data will differ if we sample (or observe) again from the true distribution. In Bayesian statistics, once observed, data itself is treated as fixed, and the model is a random variable with its own probability distribution. That is, the model uncertainty given the data does exist. Because probability is our subjective belief, prior is another distinctive concept in Bayesian statistics. How results are interpreted also differs. In Frequentist statistics, a 95% confidence interval means that if the same experiment were conducted many times, around 95% of the calculated intervals would contain the true value of the model parameter. Similarly, p-values assess how consistent the observed data is with the assumption that the null hypothesis is true. In Bayesian statistics, interpretation of probability is literally related to uncertainty.

An example of coin flipping explains Bayesian statistics well. Suppose a two-sided coin with heads and tails. The coin is fair, meaning a coin toss is equally likely to result in

heads or tails. Given a sequence of  $n$  coin flips, our goal is to determine the next coin flip. For a concrete example, we flipped the coin  $n = 3$  times. At this instance, we got 2 heads and 1 tail. Then, what is the probability of the next coin flip turning out to be heads? As a simple remedy, we can count the number of heads so far and divide by the total number of coin flips. Then, the probability estimate  $\bar{\theta}$  is given by  $\bar{\theta} = \frac{2}{3} \approx 0.667$ . The estimated probability estimate differs significantly from the true value of 0.5. This is because we naively computed the maximum likelihood estimate (MLE) of the probability of seeing a head. Indeed, a Frequentist statistician would use the data only and say  $\bar{\theta} \approx 0.667$ . Then, an error bar would be assigned to  $\bar{\theta}$ , resulting in a p-value and rejecting the hypothesis. The conclusion would not be, however, the estimated  $\bar{\theta}$  is improbable. Instead, the analysis will focus on the data being more likely to vary once the experiments are repeated.

In contrast, a Bayesian statistician would treat the probability of heads  $\theta$  as a random variable itself. As briefly introduced in chapter 1, Bayes' rule can be applied:

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})}, \quad (2.1)$$

where the data  $\mathcal{D}$  contains the history of coin flips. Again, this says that the posterior distribution of  $\theta$ , conditioned on the data, is proportional to the likelihood  $p(\mathcal{D}|\theta)$  of the data occurring given our distribution over  $\theta$  and our prior belief on  $\theta$ , denoted  $p(\theta)$ . The evidence  $p(\mathcal{D})$  normalizes the proportional term to obtain a valid probability distribution.

Starting with the likelihood term, we define the random variable  $\mathbf{x}$  to represent the number of heads. Given the probability of a head  $\theta$ , we model  $\mathbf{x}|\theta \sim \text{Binom}(n, \theta)$ :

$$P(\mathbf{x} = k|\theta) = \binom{n}{k} \theta^k (1 - \theta)^{n-k}. \quad (2.2)$$

The binomial distribution describes the number of heads  $\mathbf{x}$  being a particular  $k$  given  $\theta$ . From the above example, then  $n = 3$  and  $k = 2$ . This model extends, into sequences, a set of Bernoulli distributions – a discrete distribution of a random variable with a value of either 0 or 1 with probabilities  $\theta$  or  $1 - \theta$  respectively. For our prior, we assume  $\theta \sim \text{Beta}(\alpha, \beta)$ :

$$p(\theta) = \frac{1}{B(\alpha, \beta)} \theta^{\alpha-1} (1 - \theta)^{\beta-1}, \quad (2.3)$$

where  $B(\alpha, \beta)$  is the Beta function (Murphy, 2022). Here, the  $\alpha$  and  $\beta$  are hyperparameters that can be chosen. Once they are both one, we get a uniform distribution. It gets increasingly peaked around  $\theta = 0.5$  if we increase them with equal numbers. Lastly,

$$p(\mathbf{x}) = \int_0^1 p(\mathbf{x}|\theta) P(\theta) d\theta, \quad (2.4)$$

$$= \binom{n}{k} \frac{1}{B(\alpha, \beta)} \int_0^1 \theta^{k+\alpha-1} (1 - \theta)^{n-k+\beta-1} d\theta, \quad (2.5)$$

is our evidence term. Further simplifications involve noticing that the integrand looks similar to a beta distribution with  $\alpha' = k + \alpha$  and  $\beta' = n - k + \beta$ . Thus, we can multiply by one and utilize the property that probability distributions integrate to one:

$$p(\mathbf{x}) = \binom{n}{k} \frac{B(k + \alpha, n - k + \beta)}{B(\alpha, \beta)} \int_0^1 \frac{1}{B(k + \alpha, n - k + \beta)} \theta^{k+\alpha-1} (1 - \theta)^{n-k+\beta-1} d\theta \quad (2.6)$$

$$= \binom{n}{k} \frac{B(k + \alpha, n - k + \beta)}{B(\alpha, \beta)} \quad (2.7)$$

The evidence term is in a closed form due to our choice of the prior and the likelihood, i.e., our prior is *conjugate* to the likelihood (which cannot be assumed in practice). Finally, if we combine the prior, the likelihood, and the evidence together, we obtain the expression for the posterior (derivations are skipped as it only involves algebraic manipulation):

$$p(\boldsymbol{\theta}|\mathbf{x}) = \frac{1}{B(k + \alpha, n - k + \beta)} \boldsymbol{\theta}^{k+\alpha-1} (1 - \boldsymbol{\theta})^{n-k+\beta-1}. \quad (2.8)$$

The posterior distribution is again a beta distribution:  $\boldsymbol{\theta}|\mathbf{x} \sim \text{Beta}(k + \alpha, n - k + \beta)$ .

Now, having obtained the posterior distribution, let's try to predict the next coin flip  $\mathbf{x}^*$  given all the history of previous coin flips  $\mathbf{x}$ . Since each flip follows a Bernoulli distribution,  $p(\mathbf{x}^*|\boldsymbol{\theta}, \mathbf{x}) = \boldsymbol{\theta}$ . First, we could use the so-called maximum a-posteriori (MAP) estimate, which is a point estimate of  $\boldsymbol{\theta}$  that maximizes the posterior distribution. A major difference to the MLE estimate is the incorporation of the prior. From the definition:

$$\hat{\boldsymbol{\theta}} = \frac{k + \alpha - 1}{n + \alpha + \beta - 2}. \quad (2.9)$$

Here, if  $\alpha = \beta = 1$ , then we recover the MLE estimate. Let's set  $\alpha = \beta = 2$ . Then,  $\hat{\boldsymbol{\theta}} = 0.6$  for our considered example. This value is slightly closer to the true probability of 0.5. We note that the posterior mode is still a point estimate, throwing away other information captured in the posterior distribution. Another alternative is to look at the posterior mean and posterior variance, which integrate over the posterior distribution:

$$\mathbb{E}[\boldsymbol{\theta}|\mathcal{D}] = \frac{k + \alpha}{\alpha + \beta + n} \quad \text{and} \quad \sigma \approx \sqrt{\frac{\hat{\boldsymbol{\theta}}(1 - \hat{\boldsymbol{\theta}})}{n}}. \quad (2.10)$$

Here,  $\sigma = \sqrt{\mathbb{V}[\boldsymbol{\theta}|\mathcal{D}]}$  is the standard deviation of the posterior variance  $\mathbb{V}[\boldsymbol{\theta}|\mathcal{D}]$ . These expressions are obtained by computing expectations and variance of the posterior distribution:  $\boldsymbol{\theta}|X \sim \text{Beta}(k + \alpha, n - k + \beta)$ . Again, let's set  $\alpha = \beta = 2$ . Then,  $\mathbb{E}[\boldsymbol{\theta}|\mathcal{D}] \approx 0.571$  and  $\sigma = 0.285$ , which is closer to the true probability. We also see that uncertainty reduces at the rate of  $\frac{1}{\sqrt{n}}$  and is maximized when  $\boldsymbol{\theta} = 0.5$ . This makes sense, as one can be more sure about a biased coin. Lastly, in this example, the posterior mean corresponds to marginalization:

$$p(\mathbf{x}^*|\mathbf{x}) = \int_{\boldsymbol{\theta}} p(\mathbf{x}^*|\boldsymbol{\theta}) p(\boldsymbol{\theta}|\mathbf{x}) d\boldsymbol{\theta} \quad (2.11)$$

$$= \int_0^1 \boldsymbol{\theta} \text{Beta}(\boldsymbol{\theta}|\alpha, \beta) d\boldsymbol{\theta} = \mathbb{E}[\boldsymbol{\theta}|\mathcal{D}]. \quad (2.12)$$

Here, we have marginalized over  $\boldsymbol{\theta}$ , which is one key characteristic of Bayesian statistics.

So far, we have discussed key ideas behind Bayesian statistics. A coin flip example showed the overall procedures: (a) prior specification, (b) inference of the posterior distribution, and (c) predictions through marginalization. Next, we start from the Bayesian linear model that provides a rich view of Bayesian statistics. Then, we discuss Bayesian Neural Networks, i.e., stochastic neural networks where the model parameters are probability distributions.

### 2.3. Approximate Bayesian Inference

Approximate Bayesian inference encompasses methods that approximate the posterior distribution, when no closed form solutions exist. In the example of coin flips, the prior is conjugate to the chosen likelihood. Conjugacy enabled us to obtain the posterior distribution

in closed form. However, in more complex models typically used in practice, no conjugacy can be assumed. Then, we may rely on approximation techniques. The main topic of this thesis is to apply Bayesian statistics to deep learning. Within this context, we provide insights on the problem of approximate Bayesian inference and its challenges. To do so, we start with a linear model and discuss how its application to deep learning differs.

### 2.3.1. Linear Gauss Systems

One of the widely used probability distributions is the Multivariate normal distribution, also called the Gaussian distribution. For a random variable  $\mathbf{x} \in \mathbb{R}^d$ , it's defined as,

$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{\frac{d}{2}}|\boldsymbol{\Sigma}|^{\frac{1}{2}}} \exp\left[-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu})\right], \quad (2.13)$$

where the mean  $\boldsymbol{\mu} \in \mathbb{R}^d$  and the covariance matrix  $\boldsymbol{\Sigma} \in \mathbb{R}^{d \times d}$  parameterize the distribution. The covariance matrix is defined to be positive semi-definite  $\mathbf{v}^T \boldsymbol{\Sigma} \mathbf{v} \geq 0$  for all vectors  $\mathbf{v} \in \mathbb{R}^d$  and Hermitian, i.e., all eigenvalues are greater than or equal to zero. For linear systems, useful properties of the Gaussian distribution are closure under multiplication, linear maps, conditioning, and marginalization, which have analytical expressions.

#### Closure under Multiplication

Given two Gaussian distributions  $\mathcal{N}(\mathbf{x}|\mathbf{a}, \mathbf{A})$  and  $\mathcal{N}(\mathbf{x}|\mathbf{b}, \mathbf{B})$ , their product is given by,

$$\mathcal{N}(\mathbf{x}|\mathbf{a}, \mathbf{A})\mathcal{N}(\mathbf{x}|\mathbf{b}, \mathbf{B}) = \mathcal{N}(\mathbf{x}|\mathbf{c}, \mathbf{C})\mathcal{N}(\mathbf{a}|\mathbf{b}, \mathbf{A} + \mathbf{B}) \quad \text{where}, \quad (2.14)$$

$$\mathbf{C} = (\mathbf{A}^{-1} + \mathbf{B}^{-1})^{-1} \quad \text{and} \quad \mathbf{c} = \mathbf{C}(\mathbf{A}^{-1}\mathbf{a} + \mathbf{B}^{-1}\mathbf{b}). \quad (2.15)$$

Here,  $\mathcal{N}(\mathbf{x}|\mathbf{c}, \mathbf{C})$  is a Gaussian distribution with a normalization factor  $\mathcal{N}(\mathbf{a}|\mathbf{b}, \mathbf{A} + \mathbf{B})$ .

#### Closure under Linear Maps

Given a Gaussian marginal distribution  $p(\mathbf{x})$  and a Gaussian conditional distribution  $p(\mathbf{y}|\mathbf{x})$ , assume  $p(\mathbf{y}|\mathbf{x})$  has a mean that is a linear function of  $\mathbf{x}$  and a covariance matrix independent of  $\mathbf{x}$ . Such a linear mapping of a Gaussian distribution has an expression:

$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_x, \boldsymbol{\Sigma}_x) \quad \text{and} \quad p(\mathbf{y}|\mathbf{x}) = \mathcal{N}(\mathbf{y}|\mathbf{W}\mathbf{x} + \mathbf{b}, \mathbf{W}\boldsymbol{\Sigma}_x\mathbf{W}^T), \quad (2.16)$$

where  $\mathbf{W} \in \mathbb{R}^{k \times d}$  and  $\mathbf{b} \in \mathbb{R}^k$  are the given linear mappings from  $\mathbf{x} \in \mathbb{R}^d$  to  $\mathbf{y} \in \mathbb{R}^k$ .

#### Closure under Conditioning

The previous set up with a linear map is an example of a linear Gauss system. The corresponding distribution  $p(\mathbf{x}, \mathbf{y}) = p(\mathbf{x})p(\mathbf{y}|\mathbf{x})$  is a  $d + k$  dimensional Gaussian with:

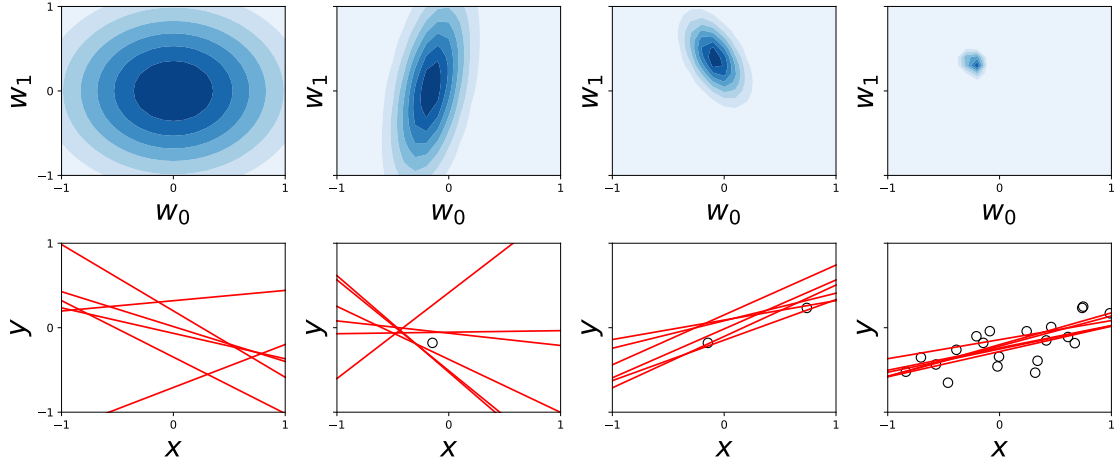
$$\boldsymbol{\mu} = \begin{pmatrix} \boldsymbol{\mu}_x \\ \mathbf{W}\boldsymbol{\mu}_x + \mathbf{b} \end{pmatrix} \quad \text{and} \quad \boldsymbol{\Sigma} = \begin{pmatrix} \boldsymbol{\Sigma}_x & \boldsymbol{\Sigma}_x\mathbf{W}^T \\ \mathbf{W}\boldsymbol{\Sigma}_x & \boldsymbol{\Sigma}_y + \mathbf{W}\boldsymbol{\Sigma}_x\mathbf{W}^T \end{pmatrix}. \quad (2.17)$$

When applying conditioning, meaning we apply Bayes' theorem to obtain  $p(\mathbf{x}|\mathbf{y})$ :

$$p(\mathbf{x}|\mathbf{y}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_{x|\mathbf{y}}, \boldsymbol{\Sigma}_{x|\mathbf{y}}) \quad \text{where} \quad (2.18)$$

$$\boldsymbol{\Sigma}_{x|\mathbf{y}}^{-1} = \boldsymbol{\Sigma}_x^{-1} + \mathbf{W}^T \boldsymbol{\Sigma}_y^{-1} \mathbf{W} \quad \text{and} \quad \boldsymbol{\mu}_{x|\mathbf{y}} = \boldsymbol{\Sigma}_{x|\mathbf{y}} [\mathbf{W}^T \boldsymbol{\Sigma}_y^{-1} (\mathbf{y} - \mathbf{b}) + \boldsymbol{\Sigma}_x^{-1} \boldsymbol{\mu}_x]. \quad (2.19)$$

This expression is called Bayes' rule for Gaussians, making it closed under conditioning.



**Figure 2.4.:** On how posterior distribution of a Bayesian linear model evolve as more and more data are added (Bishop & Nasrabadi, 2006). Top: distribution over model parameters. Bottom: data (black circles) and sampled prediction (red lines) from the posterior.

### Closure under Marginalization

Another useful property is that Gaussians are closed under marginalization:

$$p(\mathbf{y}) = \int \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_x, \boldsymbol{\Sigma}_x) \mathcal{N}(\mathbf{y}|\mathbf{W}\mathbf{x} + \mathbf{b}, \boldsymbol{\Sigma}_y) d\mathbf{x} = \mathcal{N}(\mathbf{y}|\mathbf{W}\boldsymbol{\mu}_x + \mathbf{b}, \boldsymbol{\Sigma}_y + \mathbf{W}\boldsymbol{\Sigma}_x\mathbf{W}^T) \quad (2.20)$$

A more generic description is that the Gaussian prior is conjugate to the Gaussian likelihood when the underlying process is a linear Gauss system.

### Why Gaussians for Approximate Bayesian Inference?

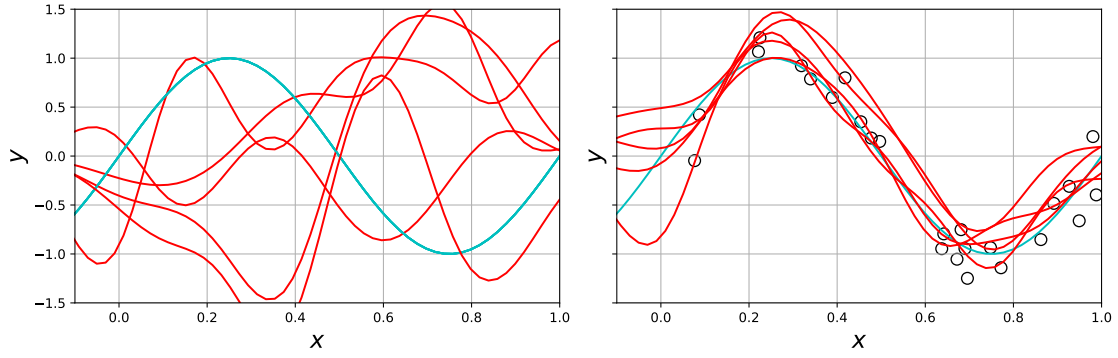
Again, Gaussian distribution is one of the go-to distributions in approximate Bayesian inference. In this thesis also, Gaussian distribution is the only choice we make for reasoning about uncertainty in deep learning. We have several reasons for this. First, the aforementioned properties show that computations behind Gaussians are linear algebra computations. Modern computers are highly specialized for this. Second, Gaussians are easy to interpret and are fully parameterized by two quantities, mean and covariance. Lastly, the central limit theorem states that sums of independent random variables can be approximated as a Gaussian distribution, making it suitable for modeling residual errors.

#### 2.3.2. Bayesian Linear Models

Many useful properties of Gaussian distributions are well exploited in Bayesian linear models. Bayesian linear models also reveal a richer view on Bayesian statistics. One exemplary case is that of regression, where given input  $\mathbf{x} \in \mathbb{R}^d$ , the goal is to obtain a linear function that predicts the target  $\mathbf{y} \in \mathbb{R}^k$ . For this, we are given a training dataset  $\mathcal{D} = (\mathbf{x}_i, \mathbf{y}_i)_{i=1}^N$ . This data is then modeled with a linear model:

$$f_{\text{linear}}(\mathbf{x}) = \mathbf{W}^T \boldsymbol{\phi}(\mathbf{x}) \quad \text{and} \quad \mathbf{y} = f_{\text{linear}}(\mathbf{x}) + \varepsilon \quad (2.21)$$

where  $\boldsymbol{\phi}(\mathbf{x})$  is a feature vector and  $\varepsilon$  is our noise model that captures some error between the model's prediction and reality. Assuming the additive noise:  $\varepsilon \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I})$  and therefore, each  $\mathbf{y}_i \sim \mathcal{N}(\mathbf{W}^T \boldsymbol{\phi}(\mathbf{x}_i), \sigma^2 \mathbf{I})$ . A stacked feature vector is denoted as  $\boldsymbol{\Phi}$ .



**Figure 2.5.:** The truth function is a sinusoidal (depicted in cyan). Left: how five samples from the prior distribution (red lines) make predictions about the true function. Right: we observe a noisy training data (black circle). After the model is obtained, we pick five samples randomly (red lines).

Let's start by defining a generic Gaussian prior over the model parameters:

$$p(\mathbf{W}) = \mathcal{N}(\mathbf{W}|\mathbf{W}_0, \mathbf{\Sigma}_0). \quad (2.22)$$

We note that if we let  $\mathbf{W}_0 = 0$  and  $\mathbf{\Sigma}_0 = \tau^2 \mathbf{I}$  (for a constant  $\tau \in \mathbb{R}$ ), the resulting prior is called zero mean isotropic prior, which is also often used. Then, we define our likelihood:

$$p(\mathcal{D}|\mathbf{W}, \sigma^2) = \prod_{i=1}^N p(\mathbf{y}_i|\mathbf{W}^T \phi(\mathbf{x}_i), \sigma^2) = \mathcal{N}(\mathbf{y}|\mathbf{\Phi}\mathbf{W}, \sigma^2 \mathbf{I}). \quad (2.23)$$

An assumption here is an I.I.D setting, enabling us to express the likelihood as a product of the Gaussian distributions. We can then apply Bayes' rule to obtain the posterior:

$$p(\mathbf{W}|\mathcal{D}, \sigma^2) \propto \mathcal{N}(\mathbf{W}|\mathbf{W}_0, \mathbf{\Sigma}_0) \mathcal{N}(\mathbf{y}|\mathbf{\Phi}\mathbf{W}, \sigma^2 \mathbf{I}) = \mathcal{N}(\mathbf{W}|\hat{\mathbf{W}}, \hat{\mathbf{\Sigma}}), \quad \text{where} \quad (2.24)$$

$$\hat{\mathbf{W}} = \hat{\mathbf{\Sigma}}(\mathbf{\Sigma}_0 \mathbf{W}_0 + \frac{1}{\sigma^2} \mathbf{\Phi}^T \mathbf{\Phi}) \quad \text{and} \quad \hat{\mathbf{\Sigma}} = (\mathbf{\Sigma}_0^{-1} + \frac{1}{\sigma^2} \mathbf{\Phi}^T \mathbf{\Phi})^{-1}. \quad (2.25)$$

Importantly, we have directly applied the property that the Gaussians are closed under conditioning. This enables us to obtain an analytical expression for the posterior.

We illustrate the procedures so far in figure 2.4. This example assumes one-dimensional input and target. The linear model is therefore a line. The prior is chosen with zero-mean isotropic Gaussian, which is depicted in the first column. Then, we gradually add new observations and apply the Bayesian modeling. For example, in the second column, we add a single data point and show how the prior is turned into the posterior. The resulting posterior now captures a positive correlation between  $b = w_0$  and  $w_1$ . For example, as the intercept  $b$  is larger, the steeper the slope  $w_1$  needs to be in order to capture the data. Since the added data point is negative, the two parameters would change in the same direction. In the third column, we added more data, which sharpens the posterior distribution. The posterior is even further sharpened in the fourth column, as more data is observed.

Now, we use the obtained posterior distribution to make predictions in a Bayesian way:

$$p(\mathbf{y}^*|\mathbf{x}^*, \mathcal{D}, \sigma^2) = \int \mathcal{N}(\mathbf{y}|\mathbf{W}^T \phi(\mathbf{x}), \sigma^2 \mathbf{I}) \mathcal{N}(\mathbf{W}|\hat{\mathbf{W}}, \hat{\mathbf{\Sigma}}) d\mathbf{W}, \quad (2.26)$$

$$= \mathcal{N}(\mathbf{y}|\hat{\mathbf{W}}^T \phi(\mathbf{x}), \sigma^2 \mathbf{I} + \phi(\mathbf{x})^T \hat{\mathbf{\Sigma}} \phi(\mathbf{x})). \quad (2.27)$$

For new test datum  $\mathbf{x}$ , we obtained a distribution over the target  $\mathbf{y}$ . The property of closure under marginalization is used to derive these expressions, where the uncertainty associated with our predictions is represented using a Gaussian distribution. We note that  $\sigma^2$  captures aleatoric uncertainty while  $\phi(\mathbf{x})^T \hat{\Sigma} \phi(\mathbf{x})$  represents epistemic uncertainty. The process of marginalization is visualized in figure 2.5. In this example, our true function is sinusoidal. We manually added some additive white noise for the observations. Note that once the training data is observed, the model is able to predict the true function. Instead of plotting the variance, we picked five samples to better illustrate the process behind marginalization. Intuitively, the insight is that the uncertainty estimates are obtained by forming multiple predictions, sampled from the posterior. These predictions tend to converge in the regime where training data is available. For the regime where training data is not available, say -0.1 to 0.1 in this example, each prediction disagrees, resulting in higher variance. When put into a continuous form instead of samples, we predict through marginalization. The variance that exists within the regimes of training data is captured using an aleatoric uncertainty term, which is irreducible even if we add more training data.

### 2.3.3. Bayesian Neural Networks

Similar procedures of Bayesian statistics can also be applied to neural networks<sup>1</sup>. However, one important difference should be noted. For non-linear models such as neural networks, the aforementioned properties of Gaussians do not hold. That is, a Gaussian prior would not be conjugate to the Gaussian likelihood, and therefore no exact closed-form solutions can be derived. As a result, the posterior and the marginalization can only be approximated.

Consider the regression problem again<sup>2</sup>. We have a data tuple  $\mathcal{D} = (\mathbf{x}_i, \mathbf{y}_i)_{i=1}^N$ . The goal is to learn a nonlinear function that predicts the target  $\mathbf{y} \in \mathbb{R}^k$  given input datum  $\mathbf{x} \in \mathbb{R}^d$ . In particular, we are interested in artificial deep neural networks:

$$f_{\theta} : \mathbb{R}^d \mapsto \mathbb{R}^k, \quad f_{\theta}(\mathbf{x}) = \varphi(\mathbf{W}_L \varphi(\mathbf{W}_{L-1} (\dots \varphi(\mathbf{W}_1 \mathbf{x}))), \quad (2.28)$$

where  $\theta = [\mathbf{W}_1, \dots, \mathbf{W}_L]$  is a vector of weights and  $\varphi(\cdot)$  is a layer-wise nonlinear function. We nest linear functions composed of nonlinearities. This type of function is also called a Multi-Layer Perceptron (MLP). Often used nonlinearity is the so-called rectified linear unit  $\text{ReLU}(\cdot) = \max(\cdot, 0)$ . Typically, these neural networks are trained using mean squared error as the loss function in regression tasks (or cross-entropy in the case of classification tasks). While neural networks are alone an interesting subject of their own, our basic description here is sufficient for us. For more interested readers, we refer to [Goodfellow et al. \(2016\)](#).

Similar to the example of coin flip (section 2.2) and Bayesian linear model (section 2.3.2), let's try to apply the procedures of Bayesian statistics to MLP. First, we specify the prior. For simplicity, we assume a zero-mean isotropic prior with precision  $\alpha$ :

$$p(\theta|\alpha) = \mathcal{N}(\theta|\mathbf{0}, \alpha^{-1} \mathbf{I}). \quad (2.29)$$

Intuitively, such types of priors are inducing regularizing effects with  $L_2$  norm so that the weights stay close to zero and fit the data less. Defining  $p(\mathbf{y}_i|\mathbf{x}_i, \theta, \beta) = \mathcal{N}(\mathbf{y}_i|f_{\theta}(\mathbf{x}_i), \beta^{-1})$ ,

$$p(\mathcal{D}|\theta, \beta) = \prod_{i=1}^N \mathcal{N}(\mathbf{y}_i|f_{\theta}(\mathbf{x}_i), \beta^{-1}) \quad (2.30)$$

---

<sup>1</sup>In this section, we provide the description with Multi-Layer Perceptrons (MLP) but the framework also applies to any neural network architectures that uses Convolution Neural Networks (CNNs), Recurrent Neural Networks (RNNs) and transformers.

<sup>2</sup>This section focuses the description to the regression problem, but Bayesian Neural Networks are applicable to any learning tasks such as classification and reinforcement learning.



is then our likelihood defined over the training data. Now, applying Bayes' rule, we know

$$p(\boldsymbol{\theta}|\mathcal{D}, \alpha, \beta) \propto p(\boldsymbol{\theta}|\alpha)p(\mathcal{D}|\boldsymbol{\theta}, \beta). \quad (2.31)$$

However, unlike the Bayesian linear model, our likelihood is characterized by a mean  $f_{\boldsymbol{\theta}}(\cdot)$ , which is a nonlinear function. As a result, Bayes' rule for Gaussians cannot be directly used. Even for zero-mean isotropic prior and a Gaussian likelihood, the problem of Bayesian inference is intractable. We have a similar problem for predictions through marginalization:

$$p(\mathbf{y}^*|\mathbf{x}^*, \mathcal{D}) = \int p(\mathbf{y}^*|\mathbf{x}^*, \boldsymbol{\theta})p(\boldsymbol{\theta}|\mathcal{D})d\boldsymbol{\theta}. \quad (2.32)$$

No closed-form expression can be obtained to evaluate the integral, which is again due to the nonlinearity induced by the neural networks, let alone the intractable posterior. These intractabilities are major differences from the linear model, which need to be approximated.

### Approximating the Posterior Distribution

One approximation technique is to locally approximate the posterior around a MAP estimate  $\hat{\boldsymbol{\theta}}$ . The MAP estimates are typically obtained through the standard training procedure in deep learning, e.g., mean squared error or cross-entropy as the loss function,  $L_2$  regularizer for initialization, and stochastic gradient descent as the optimizer. For this, we apply the second-order Taylor series approximation to the log-posterior around the MAP estimate  $\hat{\boldsymbol{\theta}}$ :

$$\ln p(\boldsymbol{\theta}|\mathcal{D}) \approx \ln p(\hat{\boldsymbol{\theta}}|\mathcal{D}) + \frac{1}{2}(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}})^T \mathbf{H}(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}). \quad (2.33)$$

Here, the term  $\mathbf{H}$  is the Hessian of the log-posterior at the MAP estimate:

$$\ln p(\boldsymbol{\theta}|\mathcal{D}) \approx \frac{\partial^2}{\partial \boldsymbol{\theta}^2} \ln p(\boldsymbol{\theta}|\mathcal{D})|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}} \quad (2.34)$$

$$= \frac{\partial^2}{\partial \boldsymbol{\theta}^2} \ln p(\mathcal{D}|f_{\boldsymbol{\theta}})|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}} + \frac{\partial^2}{\partial \boldsymbol{\theta}^2} \ln p(\boldsymbol{\theta})|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}} \quad (2.35)$$

$$=: \mathbf{H}_{\text{likelihood}} + \mathbf{H}_{\text{prior}}. \quad (2.36)$$

Thus, the Hessian is  $\mathbf{H} = -\nabla \nabla_{\boldsymbol{\theta}} \ln p(\boldsymbol{\theta}|\mathcal{D}, \alpha, \beta) = \alpha \mathbf{I} + \beta \mathbf{H}_{\text{likelihood}}$  in this case. As the Taylor series approximation is around the mode, the first-order term is assumed to have vanished. Taking the exponential on both sides, and reverse engineering the densities:

$$p(\boldsymbol{\theta}|\mathcal{D}) \approx \mathcal{N}(\boldsymbol{\theta}|\hat{\boldsymbol{\theta}}, \mathbf{H}^{-1}). \quad (2.37)$$

The Hessian captures the curvature of the loss function around the mode. The true posterior distribution is more complex than the Gaussian. But still, this technique addresses the intractable likelihood and approximates the posterior with a Gaussian distribution.

### Approximating the Predictions via Marginalization

For approximating the predictions through marginalization over model parameters, two techniques can be used. The first technique is so-called Monte Carlo integration:

$$p(\mathbf{y}^*|\mathbf{x}^*, \mathcal{D}) = \int p(\mathbf{y}^*|\mathbf{x}^*, \boldsymbol{\theta})p(\boldsymbol{\theta}|\mathcal{D})d\boldsymbol{\theta} \quad (2.38)$$

$$= \mathbb{E}_{\boldsymbol{\theta} \sim p(\boldsymbol{\theta}|\mathcal{D})} [p(\mathbf{y}^*|\mathbf{x}^*, \boldsymbol{\theta})] \quad (2.39)$$

$$\approx \frac{1}{T} \sum_{t=1}^T p(\mathbf{y}^*|\mathbf{x}^*, \boldsymbol{\theta}^{(t)}) \quad \text{for } \boldsymbol{\theta}^{(t)} \sim p(\boldsymbol{\theta}|\mathcal{D}). \quad (2.40)$$

Here,  $\theta^{(t)}$  represents a sample from the posterior. In the case of Gaussian distributions, a closed-form expression is available for sampling. In essence, the integral is evaluated by sampling from the posterior and performing multiple forward passes through the neural network using the sampled weights. An advantage of this technique is that the convergence rate is  $\frac{1}{\sqrt{T}}$  (Bishop & Nasrabadi, 2006). So, in practice, using more than 30 samples yields an acceptable approximation error. A drawback is the computational inefficiency. For a single input, we need sampling from the posterior and multiple forward passes.

Another option is to rely on a linearization scheme based on Taylor series expansion. Expanding directly on the network function around MAP estimates:

$$f_{\theta}(\mathbf{x}) \approx f_{\hat{\theta}}(\mathbf{x}) + \mathbf{J}_f^T(\theta - \hat{\theta}). \quad (2.41)$$

Here,  $\mathbf{J}_f = \nabla_{\theta} f_{\theta}(\mathbf{x})|_{\theta=\hat{\theta}}$  is the Jacobian of model parameters. Now,

$$p(\mathbf{y}|\mathbf{x}, \theta, \beta) \approx \mathcal{N}(\mathbf{y}|f_{\hat{\theta}}(\mathbf{x}) + \mathbf{J}_f^T(\theta - \hat{\theta}), \beta^{-1}), \quad (2.42)$$

because the neural network function is locally approximated as a linear model. Then,

$$p(\mathbf{y}|\mathbf{x}, \mathcal{D}, \alpha, \beta) = \mathcal{N}(\mathbf{y}|f_{\hat{\theta}}(\mathbf{x}), \beta^{-1} + \mathbf{J}_f^T \mathbf{H}^{-1} \mathbf{J}_f), \quad (2.43)$$

which uses the properties of the Gaussians. This scheme is sampling-free, i.e., unlike Monte Carlo integration, we can obtain the covariance matrix analytically. A limitation is its linearization assumption. Approximation error would not converge to zero. The scheme is also limited to regression tasks where Taylor expansion results in linearization. Therefore, for classification tasks, the so-called probit approximation can be used for both binary (Spiegelhalter & Lauritzen, 1990) and multi-class problems (Gibbs, 1998).

The described approximation techniques for the posterior and the prediction are called Laplace Approximation (or saddle-point approximation). Laplace Approximation, applied to neural networks initially (MacKay, 1992c,b), is the simplest and oldest approximate Bayesian inference technique for Bayesian Neural Networks.

### 2.3.4. Challenges Revisited

Having introduced Laplace Approximation as an approximate Bayesian inference technique for deep neural networks, we can revisit the challenges stated in chapter 1.

1. **On priors.** Prior specification is an important feature in Bayesian statistics. Through priors, we can incorporate domain knowledge into the machine learning algorithms, thereby acting as inductive bias for better generalization. Avoiding prior misspecification is also needed for the Bayesian inference to be successful. In section 2.3.3, we assumed a zero mean isotropic prior, which is also known as uninformative priors. While uninformative priors can be effective when large amounts of data is available (Fortuin, 2022), no domain knowledge can be incorporated.

A challenge is that the model parameters of neural networks are typically uninterpretable. In case of coin flip, we know that Beta distribution reasonably represent our belief about the probability of getting heads before observing a series of flips. Assume that our goal is to model the dynamics of a pendulum, and one of the random variable is an angle. We may then specify Von Mises distribution so that the angle lies within a range from 0 to  $2\pi$ . However, currently, we don't have relevant understanding about the interpretation of neural network parameters and their internal workings.

2. **On posteriors.** Posterior distribution enables us to represent our uncertainty about a model. In Bayesian statistics, the posteriors are used to derive the uncertainty estimates, especially in distributional shifts scenarios when test conditions are underrepresented in the training set. In section 2.3.3, we showed that Laplace Approximation can be used to approximate the posteriors. A key quantity is the Hessian matrix of the log-posterior, as its inverse is used as the covariance matrix.

A challenge is that the Hessian matrix is computationally intractable for modern neural network architectures and datasets. For example, ResNet (He et al., 2016) with 50 layers has about 25.632 million parameters. Forming a Hessian would require storage of 25.632 million by 25.632 million matrix, which is computationally infeasible (let alone the inversion that costs even more). To this end, approximations to the Hessian are needed. Diagonal approximation to the Hessian is popular because of computational efficiency, but it ignores the correlation between the model parameters.

3. **On predictions.** Marginalization is the key distinguishing property of a Bayesian statistics. Also known as Bayesian Model Averaging, we can combine predictions of multiple models, which are weighted by their posterior distributions. For deep neural networks, in particular, marginalization can improve accuracy and calibration, because they are under-specified by the data and different choice of hyperparameter settings yield different high performing models (Wilson, 2020; Wilson & Izmailov, 2020). Both Monte Carlo and linearization techniques are presented in section 2.3.3.

A challenge is that marginalization is computationally demanding, which can impede its applications that demand interactive runtime for the predictive models. One concrete example is robotics. For example, Monte Carlo integration (equation 2.38) involves multiple forward passes with deep neural networks. For perception tasks, where the architecture tends to be large, such mechanism can slow down the predictions. Parallelization can demand more space memory and might not be feasible if the underlying model is large. The linearization technique requires the Hessian matrix.

While we address these challenges in this thesis, it might be worth discussing why we rely on Laplace Approximation. Indeed, representing the complex posterior of deep learning with uni-modal distributions such as the Gaussians may result in crude approximations. Moreover, Laplace Approximation is also known to put symmetric probability mass around the mode (Bishop & Nasrabadi, 2006). So, in general problems, variational inference and sampling methods have usually been preferred for Bayesian inference. On the other hand, within the context of Bayesian Deep Learning, recent popularity of Laplace Approximation is motivated by several reasons. First, for deep learning, sampling methods do not scale to large architectures and datasets while variational inference is fragile (Gawlikowski et al., 2023). Thus, Laplace Approximation is an easy-to-use alternative (Daxberger et al., 2021a). Second, Laplace Approximation can be applied *post-hoc*, meaning we can take an already well-trained model (using standard training pipelines in deep learning) and approximate the Hessian of the loss function (Krause & Hübotter, 2025). Here, at least the users can obtain an accurate deep learning model and improved uncertainty estimates are an add-on. Lastly, the posterior distribution of deep neural networks is an expressive function. The distribution  $p(\theta|\mathcal{D})$  lies in high dimensional space, while its structure is provided by the underlying neural networks. We may not need to capture every detail of the posterior distribution. Gaussians in this high dimensional space are expressive functions.

We next provide an overview of the existing research areas more broadly.

## 2.4. Bayesian Deep Learning: A Brief Review

Since [Tishby et al. \(1989\)](#) proposed the idea of applying Bayesian inference to neural networks, significant advancements have been achieved. In recent years, with refurbished terminology “Bayesian Deep Learning” probabilistic view on deep learning has gained significant attention in machine learning and beyond. Bayesian Deep Learning is now an umbrella term for any Bayesian ideologies being applied to deep learning. These include not only uncertainty and extensions of Bayesian Neural Networks, but also generative deep models such as variational autoencoders, diffusion models, and normalizing flows. Kernel methods, deep Gaussian Processes, deep ensembles, and approximate Bayesian inference in large scale are other exemplary topics within this field. In this section, we classify existing works into (a) the prior specifications, (b) the inference of the posteriors, and (c) predictions through marginalization. Then, we briefly discuss existing approaches.

### 2.4.1. On Priors in Bayesian Deep Learning

Traditionally, a zero-mean isotropic prior was thought to be sufficient, and instead, the choice of the architecture introduces inductive biases into the model ([Nalisnick, 2018](#)). In recent years, however, many works report the signs of prior misspecification, leading to the so-called cold posterior effect ([Wenzel et al., 2020](#)) where a tempered posterior performs better than the high-fidelity posterior. We thereby review the recent advances on priors.

**Weight-space priors.** This type of prior is specified directly on the model parameters. An isotropic prior is one example. A natural extension to this is the full Gaussian distribution, often reduced into matrix normal distributions in which covariance matrices are Kronecker-factorized ([Louizos & Welling, 2016](#)). Inverse Wishart distribution ([Ober & Aitchison, 2021](#)), Student-t distribution ([Neklyudov et al., 2019](#)), and Laplace distribution ([Williams, 1995](#)) are other examples where the lack of expressivity in the isotropic prior is being addressed. Task-specific priors can also be found. A notable example is in the domain of model compression with Bayesian Neural Networks, for which the horseshoe priors ([Carvalho et al., 2009](#); [Louizos et al., 2017](#)) have been proposed in the past. For computer vision tasks, weight-space priors have been devised for the convolution operations ([Pearce et al., 2020](#); [Simoncelli, 2009](#)). Radial directional priors ([Farquhar et al., 2020a](#); [Oh et al., 2020](#)) are another interesting subject. The key idea is to disentangle the direction of the weight vector from its length. In this way, the learning algorithm can exploit the Goldilocks zone hypothesis ([Fort & Scherlis, 2019](#)) that states there exists an annulus of a certain length from the origin where a high density of high-performing model parameters is found.

**Function-space priors.** Another approach is to specify the prior directly on the functions rather than on model parameters. Gaussian Processes (GPs) provide a good example where priors are specified using the mean and the kernel functions ([Rasmussen, 2003](#)). When using zero mean, squared exponential kernels, like Radial Basis Functions (RBF), smoothness and the vertical scale of the functions can be controlled by the hyperparameters of the kernel. If the underlying function is periodic, the ExpSineSquared kernel can be used. For neural networks, a natural idea on function-space priors is to use GP priors and perform Bayesian inference in function-space ([Sun et al., 2019](#); [Carvalho et al., 2020](#); [Rudner et al., 2022a](#)). We can also directly fit the GP inside the neural network architecture. In this idea of deep kernel learning, a GridInterpolationKernel with an RBF base kernel is used to specify a prior over the functions ([Wilson et al., 2016](#)). Neural Tangent Kernel (a linear kernel defined by the Jacobian of neural networks ([Jacot et al., 2018](#))) is also a function-space prior, which exploits the universal approximation theorem of neural networks.

**Learning-based priors.** Typically, priors should be specified before observing data. Indeed, the idea of learning the prior seems absurd at first look. In standard settings of Bayesian inference, empirical Bayes (Bishop & Nasrabadi, 2006), which learns the hyperparameters of the prior distribution, has been discussed for Bayesian model selection. For example, approaches based on moment matching (Wu et al., 2019) and the Hessian approximations (Immer et al., 2021a) have been proposed in the past. However, the story changes if we expand the settings to sequential Bayesian inference, where the posteriors from the previous data are used as a prior for the current data. Bayesian transfer learning, continual learning, and meta learning are a few examples that exploit such a multi-task setup. For Bayesian Neural Networks, learning-based priors based on both weight-space (Schnaus et al., 2023) and function-space (Rudner et al., 2023, 2025) formulations have been proposed recently. If we consider meta learning, researchers proposed to meta-learn the BNN prior (Rothfuss et al., 2021a). These ideas have been extended to utilize PAC-Bayesian bounds (Rothfuss et al., 2021a,b, 2024), which can avoid meta-overfitting.

### 2.4.2. On Posteriors in Bayesian Deep Learning

Different methods exist for approximate Bayesian inference, which is an active area of research in machine learning and statistics. We classify the existing approaches into three categories, namely variational inference, sampling, and Laplace Approximation.

**Variational inference.** Variational inference (VI) frames the approximate Bayesian inference as an optimization problem. First, VI defines a simpler parameterized distribution called the variational family. Then, the parameters of this distribution are found by minimizing the Kullback-Leibler (KL) divergence between the variational family and the true posterior. As the true posterior is not directly accessible, VI minimizes the so-called evidence lower bound (ELBO), which approximates the KL divergence. In Bayesian Deep Learning, Hinton & van Camp (1993) applied VI to neural networks, which is then revisited by Graves (2011). When compared to the 1990s, neural networks in the modern era are deeper and use larger amounts of data. Many works that extend the early works (Hinton & van Camp, 1993; Barber & Bishop, 1998) to stochastic VI, where only a mini-batch of data is used (Graves, 2011; Blundell et al., 2015; Kingma et al., 2015). These attempts relied on mean field approximations. Frameworks with more expressive variational families, such as the matrix normal distribution, have also been investigated (Louizos & Welling Max, 2016; Osawa et al., 2019; Farquhar et al., 2020b). Perhaps the most success in VI has been achieved by reinterpreting existing stochasticity in deep learning. Monte Carlo dropout (Gal, 2016; Gal & Ghahramani, 2016) is one of the most used methods in practice. Gal & Ghahramani (2016) shows that training a neural network with dropout is an approximation of VI. Therefore, activating dropout at test time and combining the resulting stochastic predictions is an approximation of Bayesian model averaging. In this way, with minimal implementation efforts and without any expert knowledge of Bayesian inference, practitioners can reason about uncertainty in deep learning.

**Sampling methods.** Markov Chain Monte Carlo (MCMC) is a sampling method, often used when direct sampling from a distribution is difficult. A sequence of random samples, a Markov Chain where the probability of transitioning to the next state depends on the current state only, is constructed. These samples converge to the desired distribution by an acceptance criterion for the candidate samples. MCMC results in a non-parametric distribution. In the 1990s, a more efficient variant of MCMC called Hamiltonian Monte Carlo (HMC) was invented by Neal (1996b). HMC is often accepted as the gold standard of Bayesian inference (Dubey et al., 2016; Li & Gal, 2017). However, for modern architectures

and datasets, HMC is computationally infeasible (Gawlikowski et al., 2023; Li & Gal, 2017). In this context, instead of an algorithm that uses the entire dataset, stochastic extensions have been proposed (Welling & Teh, 2011; Chen et al., 2014; Dubey et al., 2016; Teh et al., 2016). These types of methods rely on stochastic gradient descent with so-called Langevin dynamics (HMC without the momentum term). Hence, they are called stochastic gradient Markov Chain Monte Carlo (SG-MCMC) methods. While SG-MCMC cannot be applied in practice due to computational inefficiency, these methods have been applied to neural networks on MNIST and CIFAR as ground truth posterior distributions. The insights gained from these analyses have led to many empirical findings on the posteriors of neural networks (Wenzel et al., 2020; Izmailov et al., 2021).

**Laplace Approximation.** Section 2.3.3 introduces Laplace Approximation, where the Hessian of neural networks is used to model a Gaussian posterior. Laplace Approximation is one of the oldest Bayesian methods that has been applied to neural networks. Early works use Laplace Approximation for uncertainty quantification (Denker & LeCun, 1991; MacKay, 1992c), Bayesian model selection (MacKay, 1992a), and also active learning (MacKay, 1992b). In recent years, noting that modern architectures tend to be larger, Ritter et al. (2018b,a) it is proposed to leverage layer-wise Kronecker factored approximations to the Hessian (Grosse & Martens, 2016; Botev et al., 2017), which are often used in the 2nd order optimization methods in deep learning. In more recent years, researchers exploited different sparse formulations (Lee et al., 2020b; Daxberger et al., 2021b; Izmailov et al., 2020) for better efficiency and uncertainty estimates. Under the name “Linearized Laplace Approximation”, the formulation of MacKay (1992b) was revisited and extended to function-space frameworks (Scannell et al., 2024; Ortega et al., 2024; Deng et al., 2022; Immer et al., 2021b; Lee et al., 2021). Meanwhile, Daxberger et al. (2021a) popularized Laplace Approximation by providing a compelling software infrastructure and empirical advantages. Popular methods called SWAG and SWA (Maddox et al., 2019) can also be viewed as another form of Laplace Approximation, where the geometry of the loss landscape is exploited. SWAG saves the trajectory of model parameters during stochastic gradient descent. Using the samples that belong to the final phase of the training, a Gaussian distribution is empirically estimated. More recent works exploit the formalisms of Riemannian geometry (Bergamin et al., 2023; Yu et al., 2024; Da Costa et al., 2025; Roy et al., 2024; Reichlin et al., 2025), which can naturally capture the true posterior locally.

### 2.4.3. On Predictions in Bayesian Deep Learning

While various methods have been proposed in the past, we limit ourselves to the so-called sampling-free uncertainty estimation, i.e., unlike Monte Carlo integration, uncertainty estimates are obtained deterministically without relying on sampling. Depending on employed strategies, we classify existing methods into three types, namely uncertainty propagation, distillations and weight sharing, and output modification.

**Uncertainty propagation.** Propagation of uncertainty, also known as error propagation, refers to a set of techniques that reason how uncertainties in input values affect the uncertainties in an output. For a linear system, as described in section 2.3.1, Gaussian distributions over the input values can be propagated with a linear transformation analytically. When the function is non-linear, approximations are typically needed through linearization. For deep learning, uncertainty propagation rules are used for sampling-free uncertainty quantification. Certain approaches reinterpret any stochastic elements in a neural network, like dropout layers, as sources of noise and propagate the stochasticity to the output (Mae et al., 2021; Shekhovtsov & Flach, 2019; Postels et al., 2019). These work



analytically to derive uncertainty propagation rules for commonly used elements of modern deep learning. Instead of utilizing already existing stochasticity in deep neural networks, controlled noises can also be added to arbitrary activations. In this space, [Yu et al. \(2021\)](#); [Schmitt & Roth \(2021\)](#) add Gaussian noise to activations of neural networks and propagate them. [Wang et al. \(2023c\)](#); [Bergna et al. \(2025\)](#) explores different GP formulations on activations. Similar ideas on the general exponential family of distributions have also been investigated ([Wang et al., 2016](#); [Ranganath et al., 2015](#); [Charpentier et al., 2022](#)).

**Model compression.** This type of method attempts to reduce the computations behind Monte Carlo sampling (or similarly ensembles of neural networks) without compromising their accuracy and uncertainty estimates. Distillation is one possible solution to model compression. A smaller model called the “student” is trained to replicate a larger and more complex model called the “teacher”. One of the early attempts was “Dark Bayesian Knowledge” which distilled predictions from SG-MCMC ([Korattikara Balan et al., 2015](#)) with a small-sized neural network. Recent works that leverage flow-based models for distillation are also found ([Park et al., 2025](#)). Although these methods can leverage the advances of deep learning, a major drawback is that the student is still a point estimate. Another direction is to make the ensembles ([Lakshminarayanan et al., 2017](#)) more efficient through model compression. We note that Monte Carlo integration essentially forms an ensemble of neural networks, where each ensemble is formed through sampling from the posteriors. In this line, one popular approach is to share the information between the ensemble members and efficiently parameterize the members ([Wen et al., 2020](#); [Laurent et al., 2023](#); [Kim et al., 2024](#)). Also, more efficient models lead to more efficient Monte Carlo sampling and ensembles. A key challenge here is ensuring the diversity.

**Output modification.** In this type of method, the last layers of neural networks are modified or augmented for uncertainty quantification. Linearization of the last layer through Laplace Approximation falls into this category. As shown in section 2.3.3, applying Taylor series expansion to the last layer provides sampling-free uncertainty estimates, which is pioneered by [MacKay \(1992b\)](#). Recent extensions exploit sparsity in deep neural networks ([Lee et al., 2020b](#); [Daxberger et al., 2021b](#); [Izmailov et al., 2020](#)) and incorporate the framework of model selection ([Antorán et al., 2022](#)) and diffusion process ([Roy et al., 2024](#)). In a function-space view, such a Bayesian linear model can be cast as GPs. This connection is theoretically formalized in [Khan et al. \(2019\)](#), couched with the theory of Neural Tangent Kernel. [Lee et al. \(2021\)](#) extends and exploits this theory in developing a practical technique for sampling-free uncertainty estimation, drawing connections to an ensemble model. Follow-ups on this direction can be found in ([Scannell et al., 2024](#); [Ortega et al., 2024](#); [Deng et al., 2022](#); [Immer et al., 2021b](#)). In a similar spirit, [Harrison et al. \(2024\)](#) instead introduces VI for only the last layer. Another line of research is distance-aware output layers. DUQ ([Van Amersfoort et al., 2020](#)) and SNGP ([Liu et al., 2020b](#)) are popular methods that utilize RBFs or GPs in the last layer. The success of these methods relies on additional inductive biases in the feature extractors that use a Jacobian penalty ([Gulrajani et al., 2017](#)) or spectral normalization ([Miyato et al., 2018](#)), respectively.

## 2.5. Benchmarks and Evaluation Protocols

Benchmarking is generally playing a crucial role in AI research. While the ultimate validation is new applications that certain methodologies enable, benchmarking in publicly available datasets tends to be more reproducible. Standardized evaluation, community-wide collaboration, identifying strengths and weaknesses within controlled experimental

<b>Tasks</b>	Classification <sup>[1-3,5,7,10-13,16-19,21-25]</sup> & Regression <sup>[1-11,13,14,16,21,23]</sup> (incl. calibration, open set recognition, distribution shifts)
<b>Downstream</b>	Reinforcement learning <sup>[1,2,8,22]</sup> , Active learning <sup>[4,7,13,18,22,23]</sup> , Adversarial attacks <sup>[3,5,23]</sup> , Bayesian optimization <sup>[8]</sup> , Language modeling <sup>[11,17]</sup> , Continual and transfer learning <sup>[12,22,24,25]</sup> , Few-shot learning <sup>[18,25]</sup> , Generative model <sup>[19]</sup> , & Pruning <sup>[15]</sup>
<b>Metrics</b>	Test LL or NLL <sup>[1,4,6-13,16-18,20-26]</sup> , ECE or Reliability diagram <sup>[7,11-13,17,18,20,22-26]</sup> , Empirical CDF for entropy <sup>[3,5,7,12,18,21,23]</sup> , AUROC <sup>[12,17-19,20,23-25]</sup> , and Brier score <sup>[16,17,18,23]</sup> & Latency <sup>[20,21,23]</sup>
<b>Dataset</b>	Toy example <sup>[1-10,14,16,19,21]</sup> , PLEX <sup>[18,25]</sup> , CO2 <sup>[1]</sup> , Sarcos & KUKA <sup>[21]</sup> , fashionMNIST <sup>[19,24,26]</sup> , UCI <sup>[1,2,4,6-11,13,16,23,26]</sup> , ImageNet <sup>[7,11,12,16-18,23]</sup> , 20newsgroup <sup>[17]</sup> , Criteo Display Advertising Challenge <sup>[17]</sup> , RETINA <sup>[18]</sup> , CLINC OOS intent detection <sup>[20,23]</sup> , CIFAR10 <sup>[3,7,12,13,16-20,22-24]</sup> , SHVN <sup>[7,16,19,20,24]</sup> , CIFAR100 <sup>[5,10,11,12,18,20,23,24,26]</sup> , Wilds <sup>[22]</sup> , Diabetic Retinopathy Detection <sup>[23]</sup> , Wikipedia Toxcity <sup>[23]</sup> , MNIST <sup>[1-3,5,7,16,17,19,22,25,26]</sup> , notMNIST <sup>[3,5,7,16]</sup> , YCB <sup>[25]</sup> & GPT2 <sup>[26]</sup> , TinyStories <sup>[26]</sup>

<sup>1</sup>(Gal & Ghahramani, 2016) <sup>2</sup>(Blundell et al., 2015) <sup>3</sup>(Louizos & Welling, 2017) <sup>4</sup>(Hernández-Lobato & Adams, 2015) <sup>5</sup>(Ritter et al., 2018b) <sup>6</sup>(Wu et al., 2019) <sup>7</sup>(Lee et al., 2020b) <sup>8</sup>(Sun et al., 2019) <sup>9</sup>(Sun et al., 2017) <sup>10</sup>(Izmailov et al., 2020) <sup>11</sup>(Maddox et al., 2019) <sup>12</sup>(Osawa et al., 2019) <sup>13</sup>(Zhang et al., 2018) <sup>14</sup>(Welling & Teh, 2011) <sup>15</sup>(Graves, 2011) <sup>16</sup>(Lakshminarayanan et al., 2017) <sup>17</sup>(Ovadia et al., 2019) <sup>18</sup>(Tran et al., 2022) <sup>19</sup>(Van Amersfoort et al., 2020) <sup>20</sup>(Liu et al., 2020b) <sup>21</sup>(Lee et al., 2021) <sup>22</sup>(Daxberger et al., 2021a) <sup>23</sup>(Nado et al., 2021) <sup>24</sup>(Rudner et al., 2023) <sup>25</sup>(Schnaus et al., 2023) <sup>26</sup>(Bergna et al., 2025)

**Table 2.2.:** An overview of current benchmarks and evaluation protocols.

conditions, and tracking progress over time are other benefits of benchmarks on publicly available datasets. The keyword is a common testing ground for all the researchers. Another aspect of the benchmark is the evaluation protocol, which includes considerations on datasets, tasks at hand, baselines, and the design of the experimental conditions. Because uncertainty estimates represent a subjective measure of a random quantity, no ground truth exists. This makes the evaluation of uncertainty quantification methods challenging. A summary of the current practices in existing works is summarized in table 2.2. We note that more recent literature is incorporated when compared to existing surveys (Gawlikowski et al., 2023). Learning tasks, datasets, and typical metrics are detailed in the table.

In current literature, different methods are evaluated based on how reliable uncertainty estimates are, while keeping the predictions accurate and efficient. While metrics for accuracy and efficiency are well known, evaluation criterion on uncertainty has been an evolving quantity. One popular approach is to rely on the so-called proper score rules (Gneiting & Raftery, 2007). Scoring rules measure how reliable a predictive distribution  $p(\mathbf{y}^*|\mathbf{x}^*, \mathcal{D})$  is. By definition, given a data tuple  $(\mathbf{y}, \mathbf{x})$ ,  $S(p(\mathbf{y}^*|\mathbf{x}^*, \mathcal{D}), q) = \int q(\mathbf{y}, \mathbf{x})S(p(\mathbf{y}^*|\mathbf{x}^*, \mathcal{D}), (\mathbf{y}, \mathbf{x}))d\mathbf{x}d\mathbf{y}$  is an expected scoring rule where  $q(\mathbf{y}, \mathbf{x})$  denotes the true distribution. Then, a proper scoring



rule is defined as,  $S(p(\mathbf{y}^*|\mathbf{x}^*, \mathcal{D}), q) \leq S(q, q)$  with equality if and only if  $p(\mathbf{y}^*|\mathbf{x}^*, \mathcal{D}) = q(\mathbf{y}|\mathbf{x})$ . Log likelihood (at test set and similarly negative log likelihood) and Brier score are often used proper scores. Another metric is Expected Calibration Error (ECE) (Guo et al., 2017). The calibration error measures the gap between the accuracy – how many predictions in a group (bins from discretizations) are actually correct – and the average predicted probability (confidence) for that group. The ECE is the average of these differences. Area Under Receiver Operating Characteristic curve (AUROC) and Area Under Precision-Recall curve (AUPR) (Hendrycks & Gimpel, 2017) are also often used, which measure how well uncertainty estimates can separate correct and wrong predictions. Empirical CDF for entropy (Louizos & Welling, 2017) is a similar metric to AUROC. Indeed, uncertainties cannot be directly measured. So, the key insight behind these metrics is whether uncertainty estimates align with both correctness and incorrectness of the predictions.

Typically, these metrics are used under certain experimental protocols. These protocols create situations where the underlying predictor is both correct and incorrect. The certainty of a predictive distribution is evaluated under I.I.D settings, where test conditions resemble that of training (a typical train and test set within one dataset). Uncertainty estimates are then evaluated when test conditions differ from the training setup. A typical example is open-set recognition (Tran et al., 2022), which assesses how well a model can detect test examples not belonging to the training data. For example, a model can be trained on MNIST and tested on notMNIST or FashionMNIST. In this case, uncertainty estimates should be high when tested on notMNIST or FashionMNIST. Another pair is CIFAR10 and SHVN. While the protocol of open-set recognition creates out-of-distribution conditions for the model, test conditions can also slightly differ from the training. Distribution shifts refer to scenarios where the distribution of test conditions slightly changes in such a way that the model still fails (Ovadia et al., 2019). Lastly, downstream tasks of Bayesian models are also often used. These protocols evaluate how well the downstream tasks are solved. Examples span many tasks such as reinforcement learning, active learning, Bayesian optimization, continual and transfer learning, and many more. These downstream tasks are designed so that better uncertainty estimates result in higher performance in tackling these tasks.

An interesting trend can be observed by looking at the typical datasets in uncertainty benchmarks. In the early days, datasets from the UCI repository (Dua & Graff, 2017) have been widely adopted. Created in 1987, the dataset consists of more than 500 datasets, often involving statistics problems. UCI datasets are often small-scale and simple. Therefore, the datasets offer an advantage that algorithmic intuitions are easier to capture than complex real-world datasets. However, the empirical findings from the datasets may not transfer to real-world applications. MNIST and CIFAR are another exemplary datasets (slightly bigger than most UCI datasets), which are used to build intuitions and fast experimentation. From table 2.2, we can see that these datasets are still used today. However, we also see the trend that more challenging datasets are also used in parallel. Some examples are ImageNET, which constitutes over 14 million images for the recognition of 1000 classes. ImageNET is often used to demonstrate the scalability of Bayesian inference methods. The WILDS repository (Koh et al., 2021) also provides an interesting benchmark. The repository consists of 10 multi-modal datasets, which capture real-world distribution shifts. In more recent years, language datasets such as GPT-2 are also being used, addressing the hallucination problems of large language models (Bergna et al., 2025).

Benchmark and evaluation protocol are evolving as progress is being made. Ovadia et al. (2019); Tran et al. (2022); Nado et al. (2021) contributes to more considerations on how we can benchmark different uncertainty quantification methods.

## 2.6. Summary

So far, we have provided a brief introduction to uncertainty in robotics and deep learning. Examples have been provided to show that robots encounter uncertainties which are irreducible and exist all the time (“aleatoric uncertainty”). Then, we discussed uncertainties that arise due to unseen situations (“epistemic uncertainty”). This thesis leverages Bayesian statistics for the given problem. Background on Bayesian statistics is presented with an example of coin flips. Then, we discussed Bayesian inference and presented Bayesian linear models and Bayesian Neural Networks. We also listed the challenges of Bayesian inference for deep learning and provided a brief review on the current state of the art. Benchmarking and measuring the quality of uncertainty estimates is an important topic, especially for comprehending scientific experiments on uncertainty in deep learning. To this end, we provided a meta-analysis on current practices on benchmarking uncertainty estimates.

## Part II.

# Bayesian Algorithms for Deep Neural Networks

Applying Bayesian statistics to deep learning is a challenging endeavor. A default choice of prior has been zero-mean isotropic Gaussian despite the reported signs of prior specification. Mean field approximations or even Bernoulli distributions are often used to represent the posteriors despite the lack of expressivity. Marginalization, often solved by Monte Carlo integration techniques, relies on sampling and is inefficient in terms of run-time. While encouraging progress has been accomplished in the past, most of the algorithms have been evaluated on UCI, MNIST, and CIFAR (as discussed in chapter 2). As a result, there may not be convincing evidence yet that these methods are ready to be deployed in practice. In the following chapters, for each of the aforementioned challenges, we therefore propose new methods that are suited for their applications in robotics. We draw inspiration from the theory of generalization, information, and the universal approximation theorem of neural networks. These methods are also benchmarked against existing approaches while we further analyze these methods theoretically using first principles and mathematics.



---

On Prior Specification for Bayesian Neural Networks

---

### 3.1. Introduction

Within the deep learning approach to real-world AI problems such as autonomous driving, generalization and uncertainty estimation are one of the most important pillars. To achieve this, Bayesian Neural Networks (BNNs) (MacKay, 1992c; Hinton & van Camp, 1993; Neal, 1996b) leverage the tools of Bayesian statistics to improve generalization and uncertainty estimation in deep neural networks. Due to their potential and advances so far, BNNs have become increasingly popular research topics (Gawlikowski et al., 2023). However, one of the open problems in BNNs is the prior specifications. Although prior selection is widely known to be a crucial step in any Bayesian modeling (Bayes, 1763), the choice of well-specified priors is generally unknown in neural networks. Consequently, many current approaches have resorted to uninformative priors, such as isotropic Gaussian, despite the reported signs of prior misspecifications (Wenzel et al., 2020; Fortuin, 2022).

To address the problem of prior specification, we propose a prior learning method for BNNs. Our method is grounded in sequential Bayesian inference (Oppen, 1999), where posteriors from the past are used as the prior for the future. We achieve this relying on the Laplace approximation (LA) (MacKay, 1992c) with Kronecker factorization of the posteriors (Ritter et al., 2018b; Daxberger et al., 2021a). Within this framework, we devise key technical novelties to learn expressive BNN priors with generalization guarantees. First, we demonstrate the sums-of-Kronecker-product computations so that the posteriors in matrix normal distributions, i.e., Gaussian with Kronecker-factorized covariance matrices (Gupta & Nagar, 2000), can be tractably used as expressive BNN priors. Second, to explicitly improve generalization, we derive and optimize over a tractable approximation of PAC-Bayes bounds (McAllester, 1999b; Germain et al., 2016) that lead to non-vacuous bounds, i.e., smaller than the upper bound of the loss. Finally, as an added benefit of our idea, a Bayesian reinterpretation of a popular continual learning architecture, namely progressive neural networks (Rusu et al., 2016), is provided for awareness of uncertainty, generalization, and resilience to catastrophic forgetting.

By design, our method has many advantages for generalization and uncertainty estimation in neural networks. The proposed method achieves non-vacuous generalization bounds in deep learning models, while potentially avoiding prior misspecification for uncertainty

estimation using BNNs. For these features, we provide computationally tractable techniques in order to learn expressive priors from large amounts of data and deep network architectures. We contend that such probabilistic representation at a large scale, e.g., pre-trained BNNs on ImageNet, can be beneficial in downstream tasks for both transfer and continual learning set-ups. Therefore, we also provide ablation studies and various experiments to show the aforementioned benefits of our method within the small- and large-scale transfer and continual learning tasks. In particular, we empirically demonstrate that our priors mitigate cold posterior effects (Wenzel et al., 2020) – a potential sign of a bad prior – and further produce generalization bounds. Moreover, within a practical scenario of robotic continual learning (Denninger & Triebel, 2018), and standardized benchmarks of few-shot classification (Tran et al., 2022) for the recent uncertainty baselines (Nado et al., 2021), considerable performance improvements over the popular methods are reported.

**Contributions and major claims.** To summarize, our primary contribution is a novel method for learning scalable and structured informative priors (section 3.2), backed up by (a) the sums-of-Kronecker-product computations for computational tractability (section 3.2.2), (b) derivation and optimizations over tractable generalization bounds (section 3.2.3), (c) a Bayesian re-interpretation of progressive neural networks for continual learning (section 3.2.4), (d) exhaustive experiments to show the effectiveness of our method, which indicate that our method can induce meaningful BNN priors (section 3.4). Theoretically, we show that the sums-of-Kronecker-product computations converge to an optimal rank-one solution. Within experimental conditions where PAC-Bayes bounds can be empirically evaluated, our method can achieve non-vacuous bounds while mitigating the signs of prior misspecification. Lastly, our method achieves state-of-the-art performance in the few-shot uncertainty benchmark, where we utilize the pretrained models on ImageNet.

## 3.2. Learning Expressive Priors

### 3.2.1. Notation and Background

Consider a neural network  $f_{\theta}$  with layers  $l \in [L] := \{1, \dots, L\}$  which consists of a parameterized function  $\phi_l : \Omega_{l-1} \rightarrow \Omega_l$  and a non-linear activation function  $\sigma_l : \Omega_l \rightarrow \Omega_l$ . We denote an input for a network as  $\mathbf{x} \in \Omega_0 =: \mathcal{X}$  and all learnable parameters as a stacked vector  $\theta$ . Then the pre-activation  $\mathbf{s}_l$  and activation  $\mathbf{a}_l$  are recursively applied with  $\mathbf{a}_0 = \mathbf{x}$ ,  $\mathbf{s}_l = \phi_l(\mathbf{a}_{l-1})$  and  $\mathbf{a}_l = \sigma_l(\mathbf{s}_l)$ . The output of the neural network  $\mathbf{y}$  is given by the last activation  $\mathbf{y} = \mathbf{a}_L = f_{\theta}(\mathbf{x})$ . The structure of the learning process is governed by different architectural choices, such as fully connected, recurrent, and convolution layers (Goodfellow et al., 2016). Parameters  $\theta$  are typically obtained by maximum likelihood principles, given training data  $\mathcal{D} = ((\mathbf{x}_i, \mathbf{y}_i))_{i=1}^N$ . Unfortunately, these principles lead to the lack of generalization and calibrated uncertainty in neural networks (Jospin et al., 2020).

To address these, BNNs provide a compelling alternative by applying Bayesian principles to neural networks. In BNNs, the first step is to specify a prior distribution  $\pi(\theta)$ . Then the Bayes theorem is used to compute the posterior distribution over the parameters, given the training data:  $p(\theta|\mathcal{D}) \propto \pi(\theta) \prod_{i=1}^N p(\mathbf{y}_i|\mathbf{x}_i, f_{\theta})$ . This means that each parameter is not a single value, but a probability distribution, representing the uncertainty of the model (Gawlikowski et al., 2023). The posterior distribution provides a set of plausible model parameters, which can be marginalized to compute the predictive distributions of a new sample  $(\mathbf{x}^*, \mathbf{y}^*) \notin \mathcal{D}$  for BNNs:  $p(\mathbf{y}^*|\mathbf{x}^*, \mathcal{D}) = \int p(\mathbf{y}^*|\mathbf{x}^*, f_{\theta})\rho(\theta)d\theta$ . In the case of neural networks, the posteriors cannot be obtained in a closed form. Hence, many of the current

research efforts focus on accurately approximating the posteriors of BNNs.

For this, LA (MacKay, 1992c) approximates the BNN posteriors with Gaussian distributions around a local mode (more details on LA are found in section 2.3.3). Here, a prior is first specified as an isotropic Gaussian. With this prior, the maximum a posteriori (MAP) estimates of the parameters  $\hat{\theta}$  are obtained by training neural networks. Then, the Hessian  $\mathbf{H} = \frac{d^2}{d\theta^2} \ln p(\theta|\mathcal{D}) = \mathbf{H}_{\text{likelihood}} + \mathbf{H}_{\text{prior}}$  is computed to obtain the BNN posteriors. In practice, the covariance matrix is usually further scaled, which more generally corresponds to temperature scaling. In summary, the priors-posteriors pairs of LA are:

$$\text{prior: } \pi(\theta) = \mathcal{N}(\mathbf{0}, \gamma \mathbf{I}) \text{ and posterior: } p(\theta|\mathcal{D}) \approx \mathcal{N}(\hat{\theta}, (\mathbf{H}_{\text{likelihood}} + \mathbf{H}_{\text{prior}})^{-1}). \quad (3.1)$$

As Hessian is computationally intractable for modern architectures, the Kronecker-Factored Approximate Curvature (KFAC) is widely adopted (Martens & Grosse, 2015). KFAC approximates the true Hessian by a layerwise block diagonal matrix, where each diagonal block is the Kronecker-factored Fisher matrix  $\mathbf{F}_l$ . This means each layer is assumed to be independent. Now, defining the layer-wise activation  $\bar{\mathbf{a}}_l$  and loss gradient w.r.t pre-activation  $\mathcal{D}\mathbf{s}_l$ , the inverse covariance matrix of the posteriors is (Ritter et al., 2018b):

$$\mathbf{H} \approx \mathbf{F} = \text{diag}(\mathbf{F}_1, \mathbf{F}_2, \dots, \mathbf{F}_L) + \gamma \mathbf{I}, \text{ where } \mathbf{F}_l \approx \mathbb{E}[\bar{\mathbf{a}}_l(\bar{\mathbf{a}}_l)^T] \otimes \mathbb{E}[\mathcal{D}\mathbf{s}_l(\mathcal{D}\mathbf{s}_l)^T] = \mathbf{A}_l \otimes \mathbf{G}_l. \quad (3.2)$$

In this way, BNN posteriors can be obtained by approximating the Hessian with KFAC.

We note several ramifications of this formulation for learning BNN priors from data. First, BNN posteriors can be obtained by approximating the Hessian, which can be tractably computed from large amounts of data and deeper neural network architectures (Lee et al., 2020b; Ba et al., 2016). Second, the resulting BNN posteriors can capture the correlations between the parameters within the same layer (Ritter et al., 2018b). Moreover, there are several open-source libraries that facilitate implementations (Daxberger et al., 2021a; Humt et al., 2020). All these points result in easily obtainable BNN posteriors with expressive probabilistic representation from large amounts of data, deep architectures, and parameter correlations. As opposed to isotropic Gaussian, we next demonstrate that these BNN posteriors can be used to learn expressive prior distributions in order to advance generalization and uncertainty estimation within the transfer and continual learning setups.

### 3.2.2. Prior Learning with Sums-of-Kronecker-Product Computations

Intuitively, the idea is to repeat the LA with the prior chosen as the posterior from 3.1, similar to Bayesian filtering. Since we use the LA with a Kronecker-factored covariance matrix in both the prior and the posterior, we want to approximate the resulting sum-of-Kronecker-products with a single Kronecker product. We denote the task to compute the prior as source task  $\mathfrak{T}_0$  and the task to compute the posterior as target task  $\mathfrak{T}_1$ , both consisting of a dataset  $\mathcal{D}_t$  from a distribution  $P_t$ ,  $t \in \{0, 1\}$ .

Applying LA on  $\mathcal{D}_0$ , we obtain the model parameters  $\hat{\theta}^{(0)}$  and the posteriors  $p(\theta|\mathcal{D}_0)$  as before. Then, for the target task, we specify the prior using the posteriors from  $\mathcal{D}_0$ :  $\pi(\theta) = p(\theta|\mathcal{D}_0)$ . The ultimate goal is to compute the new posteriors in  $\mathcal{D}_1$ , i.e.,  $p(\theta|\mathcal{D}_1) \propto p(\mathcal{D}_1|f_\theta)\pi(\theta)$ . To achieve this, we train the neural networks on  $\mathcal{D}_1$  by optimizing:

$$\begin{aligned} \hat{\theta}^{(1)} &\in \arg \max_{\theta} \{p(\theta|\mathcal{D}_1)\} \\ &= \arg \min_{\theta} \left\{ -\ln(p(\mathcal{D}_1|f_\theta)) + \frac{1}{2}(\theta - \hat{\theta}^{(0)})^T (\mathbf{F}^{(0)} + \gamma \mathbf{I})(\theta - \hat{\theta}^{(0)}) \right\}, \end{aligned}$$

```

Input : Left matrices  $(\mathbf{A}^k)_{k \in [K]}$ , right matrices  $(\mathbf{G}^k)_{k \in [K]}$ ,
          number of steps  $n^{max} = 100$ , and stopping precision  $\delta = 10^{-5}$ .
Output: The solutions  $\hat{\mathbf{A}}$  and  $\hat{\mathbf{G}}$ .

begin
    /* Initialization */
    vec( $\bar{\mathbf{A}}^{(0)}$ )  $\leftarrow \mathcal{N}(\mathbf{0}, \mathbf{I})$  ; // initialization of  $\bar{\mathbf{A}}^{(0)}$ 
     $\mathbf{A}^{(0)} \leftarrow \frac{\bar{\mathbf{A}}^{(0)}}{\|\bar{\mathbf{A}}^{(0)}\|_F}$  ; // normalize  $\bar{\mathbf{A}}^{(0)}$ 

    /* Main Loop */
    for  $n = 1$  to  $n^{max}$  do
         $\bar{\mathbf{G}}^{(n)} \leftarrow \sum_{k=1}^K \langle \mathbf{A}^k, \mathbf{A}^{(n-1)} \rangle_F \mathbf{G}^k$  ; // first power step
         $\mathbf{G}^{(n)} \leftarrow \frac{\bar{\mathbf{G}}^{(n)}}{\|\bar{\mathbf{G}}^{(n)}\|_F}$  ; // normalize  $\bar{\mathbf{G}}^{(n)}$ 
         $\bar{\mathbf{A}}^{(n)} \leftarrow \sum_{k=1}^K \langle \mathbf{G}^k, \mathbf{G}^{(n)} \rangle_F \mathbf{A}^k$  ; // second power step
         $\mathbf{A}^{(n)} \leftarrow \frac{\bar{\mathbf{A}}^{(n)}}{\|\bar{\mathbf{A}}^{(n)}\|_F}$  ; // normalize  $\bar{\mathbf{A}}^{(n)}$ 
        if  $\|\mathbf{A}^{(n)} - \mathbf{A}^{(n-1)}\|_F < \delta$  then // stopping criterion
            break ;
        end
    end
     $\hat{\mathbf{A}} \leftarrow \mathbf{A}^{(n)}$  ; // first power step
     $\hat{\mathbf{G}} \leftarrow \sum_{k=1}^K \langle \mathbf{A}^k, \mathbf{A}^{(n)} \rangle_F \mathbf{G}^k$  ; // first power step
end
    
```

Algorithm 1: Power method for sums of Kronecker products

where  $\hat{\boldsymbol{\theta}}^{(1)}$  is the MAP estimate of the model parameters for task  $\mathfrak{T}_1$  and  $\mathbf{F}^{(0)}$  is the block-diagonal Kronecker-factored Fisher matrix from task  $\mathfrak{T}_0$ . The final step is to approximate the new Hessian on  $\mathcal{D}_1$  using the KFAC method. This results in the new pairs:

$$\text{prior: } \pi = \mathcal{N}(\hat{\boldsymbol{\theta}}^{(0)}, (\mathbf{F}^{(0)} + \gamma \mathbf{I})^{-1}) \text{ and posterior: } \rho = \mathcal{N}(\hat{\boldsymbol{\theta}}^{(1)}, \tau(\mathbf{F}^{(1)} + \mathbf{F}^{(0)} + \gamma \mathbf{I})^{-1}), \quad (3.3)$$

where  $\tau$  is the temperature scaling (Wenzel et al., 2020; Daxberger et al., 2021a). Here, the precision matrix of the new BNN posteriors is represented by the sum-of-Kronecker-products, i.e.,  $\tilde{\mathbf{F}}^{(1)} = \mathbf{F}^{(1)} + \mathbf{F}^{(0)} + \gamma \mathbf{I} = \mathbf{A}^{(0)} \otimes \mathbf{G}^{(0)} + \mathbf{A}^{(1)} \otimes \mathbf{G}^{(1)} + \gamma \mathbf{I} \otimes \mathbf{I}$ .

Unfortunately, the sum-of-Kronecker-products is not a Kronecker-factored matrix. As a result, the above formulation loses all the essence of Kronecker factorization for modeling high-dimensional probability distributions. For example, we can no longer exploit the rule:  $(\mathbf{A} \otimes \mathbf{G})^{-1} = (\mathbf{A}^{-1} \otimes \mathbf{G}^{-1})$  where  $\mathbf{A}$  and  $\mathbf{G}$  are smaller matrices to store and invert when compared to  $(\mathbf{A} \otimes \mathbf{G})$ . In addition, due to their size, the storage and the inverse of  $\tilde{\mathbf{F}}^{(1)}$  may not be computationally tractable for modern architectures.

To this end, we devise the sum-of-Kronecker-products computations. For this, we propose an optimization formulation that approximates the sum-of-Kronecker-products as Kronecker-factored matrices:<sup>12</sup>

$$\hat{\mathbf{A}}, \hat{\mathbf{G}} \in \arg \min_{\substack{\mathbf{A} \in \mathbb{R}^{M \times M} \\ \mathbf{G} \in \mathbb{R}^{N \times N}}} \left\| \sum_{k=1}^K \mathbf{A}^k \otimes \mathbf{G}^k - \mathbf{A} \otimes \mathbf{G} \right\|_F. \quad (3.4)$$

<sup>1</sup>For the similar causes to maintain the Kronecker factorization, Ritter et al. (2018b,a) assume  $(\mathbf{A} \otimes \mathbf{G} + \gamma \mathbf{I})^{-1} = (\mathbf{A} + \gamma \mathbf{I})^{-1} \otimes (\mathbf{G} + \gamma \mathbf{I})^{-1}$  which does not hold in general (see section 3.4.1).

<sup>2</sup>Our formulation for one single Kronecker-factored matrix is similar to Kao et al. (2021).



A solution to this problem is not unique; for example, one could scale  $\hat{\mathbf{A}}$  by  $\alpha \neq 0$  and  $\hat{\mathbf{G}}$  by  $\frac{1}{\alpha}$ . Hence, we assume that  $\hat{\mathbf{A}}$  is normalized,  $\|\hat{\mathbf{A}}\|_F = 1$  (or alternatively, we can also assume  $\|\hat{\mathbf{G}}\|_F = 1$ ). For the solution, we show the equivalence to the well-known best rank-one approximation problem, which can be solved with the power method. When the solution matrix has rank one, the product of two vectors can express the solution.

**Lemma 3.2.1.** *Let  $M, N, K \in \mathbb{N}$ ,  $\mathbf{A}^k \in \mathbb{R}^{M \times M}$  and  $\mathbf{G}^k \in \mathbb{R}^{N \times N}$  for  $k \in [K]$ . Then*

$$\left\| \sum_{k=1}^K \mathbf{A}^k \otimes \mathbf{G}^k - \mathbf{A} \otimes \mathbf{G} \right\|_F = \left\| \sum_{k=1}^K \text{vec}(\mathbf{A}^k) \text{vec}(\mathbf{G}^k)^T - \text{vec}(\mathbf{A}) \text{vec}(\mathbf{G})^T \right\|_F. \quad (3.5)$$

*Proof.* The proof can be found in A.3.2.  $\square$

Again, this result indicates that the solution to equation 3.4 can be obtained using the power method, which iterates:

$$\mathbf{G}^{(n)} \leftarrow \frac{\sum_{k=1}^K \langle \mathbf{A}^k, \mathbf{A}^{(n-1)} \rangle_F \mathbf{G}^k}{\left\| \sum_{k=1}^K \langle \mathbf{A}^k, \mathbf{A}^{(n-1)} \rangle_F \mathbf{G}^k \right\|_F} \quad \text{and} \quad \mathbf{A}^{(n)} \leftarrow \frac{\sum_{k=1}^K \langle \mathbf{G}^k, \mathbf{G} \rangle_F \mathbf{A}^k}{\left\| \sum_{k=1}^K \langle \mathbf{G}^k, \mathbf{G} \rangle_F \mathbf{A}^k \right\|_F}.$$

Given randomly initialized matrices, the power method iterates until the stop criterion is reached. The complete procedure is presented in algorithm 1, while in appendix A.2, we discuss the computational complexity of that algorithm.

Importantly, we can now obtain a single Kronecker factorization from the sum-of-Kronecker-products. Such Kronecker-factored approximations have advantages in tractable memory consumption for the storage and inverse computations, which enable sampling from the resulting matrix normal distributions (Martens & Grosse, 2015). Therefore, with its connection to equation 3.3, such computations allow us to learn expressive BNN priors from the posterior distributions of previously encountered tasks. Finally, we can also prove the convergence of the power method to an optimal rank-one solution.

**Lemma 3.2.2.** *Let  $\mathbf{M} = \sum_{k=1}^K \text{vec}(\mathbf{A}^k) \text{vec}(\mathbf{G}^k)^T$  and  $\mathbf{M} = \sum_{i=1}^r \sigma_i \mathbf{u}_i \mathbf{v}_i^T$  be its singular value decomposition with  $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_r > 0$  and  $\mathbf{u}_i^T \mathbf{u}_j = \mathbf{v}_i^T \mathbf{v}_j = \mathbb{1}[i = j]$ . Then there is a solution of equation 3.4 with  $\text{vec}(\hat{\mathbf{A}}) = \mathbf{u}_1, \text{vec}(\hat{\mathbf{G}}) = \sigma_1 \mathbf{v}_1$ . If  $\sigma_1 > \sigma_2$ , the solution is unique up to changing the sign of both factors, and our power method converges almost surely to this solution.*

*Proof.* The proof can be found in A.3.3.  $\square$

### 3.2.3. Derivations and Optimizations over Tractable PAC-Bayes Bounds

So far, we have presented a method for learning a scalable and structured prior. This section shows how we can explicitly minimize the generalization bounds with the learned prior. We achieve this by adapting existing generalization bounds for LA. In particular, our approach allows tuning the hyperparameters of the LA on the training set without a costly grid search. For this, we derive a differentiable cost function that approximates the generalization bounds. We choose to optimize generalization bounds to trade off the broadness of the distribution and the performance on the training data to improve generalization.

A common method in BNNs, and especially in LA, is to scale the covariance matrix by a positive scalar called the temperature  $\tau > 0$  (Wenzel et al., 2020; Daxberger et al., 2021a). In addition, often the Hessian (or Fisher matrix) of the likelihood (Ritter et al., 2018b; Lee

et al., 2020b) or the prior (Gawlikowski et al., 2023) is scaled. Including all these terms, the posterior has the following form:

$$\rho = \mathcal{N}(\hat{\boldsymbol{\theta}}^{(1)}, \tau(\beta \mathbf{F}^{(1)} + \alpha \tilde{\mathbf{F}}^{(0)})^{-1}), \quad (3.6)$$

where  $\tilde{\mathbf{F}}^{(0)} = \mathbf{F}^{(0)} + \gamma \mathbf{I}$  is the precision matrix of the prior. This reweighting of the three scalars allows us to cope with misspecified priors (Wilson & Izmailov, 2020), approximation errors in Fisher matrices, and to improve the broadness of the distribution. Although these values are usually hyperparameters that have to be manually scaled by hand, we argue that these values should be application-dependent. Therefore, we propose to find all three scales –  $\alpha$ ,  $\beta$ , and  $\tau$  – by minimizing generalization bounds.

In the following, we will first introduce our two approaches and then explain how the optimization of PAC-Bayes bounds can be made tractable for the LA.

**Method 1: Curvature scaling.** *Previous approaches typically scale only one or two of the scales (Daxberger et al., 2021a; Ritter et al., 2018b; Lee et al., 2020b). With our automatic scaling, we can optimize all three scales for each individual layer, allowing the model to decide the weighting in a more fine-grained way. We can use this method not only to scale the posterior, but also to compute the prior. Thus, the prior and posterior are:*

$$\pi = \mathcal{N}(\hat{\boldsymbol{\theta}}^{(0)}, (\tilde{\mathbf{F}}^{(0)})^{-1}), \quad \rho = \mathcal{N}(\hat{\boldsymbol{\theta}}^{(1)}, (\tilde{\mathbf{F}}^{(1)})^{-1}), \quad (3.7)$$

where the diagonal blocks of the precision matrix corresponding to each layer are defined as  $\tilde{\mathbf{F}}_l^{(0)} = \frac{1}{\tau_l^{(0)}}(\beta_l^{(0)} \mathbf{F}_l^{(0)} + \alpha_l^{(0)} \gamma \mathbf{I})$  and  $\tilde{\mathbf{F}}_l^{(1)} = \frac{1}{\tau_l^{(1)}}(\beta_l^{(1)} \mathbf{F}_l^{(1)} + \alpha_l^{(1)} \tilde{\mathbf{F}}_l^{(0)})$ , respectively.

*Remark 3.2.3.* At first look, the temperature scaling could be captured within the other curvature scales, and thus the formulation seems redundant. However, we empirically find it easier to optimize the bounds with all three parameters per layer.

**Method 2: Frequentist projection.** *Going one step further, we propose scaling not only the curvature but also the network parameters using PAC-Bayes bounds. Since the Fisher matrix is known only after training, we assume the same covariance for the posterior as for the prior when optimizing the network parameters. Since this method does not optimize for the MAP parameters, but only for minimizing the PAC-Bayes bounds, we call it Frequentist projection. We note that the PAC-Bayes is not fully Bayesian.*

Both methods aim to improve the generalization of our model. To do this, we modify the PAC-Bayes bounds (Germain et al., 2016; Guedj, 2019) to derive a tractable objective. Our concrete goal is an objective that can be optimized without going through the entire training dataset. The main idea of the PAC-Bayes theory is to upper bound the expected loss in the true data distribution  $P^N$  with high probability. The bounds typically depend on the expected empirical loss on the training data and the KL-divergence between a data-independent prior and the posterior distribution. For  $\varepsilon > 0$ , that is

$$P_{\mathcal{D} \sim P^N}(\forall \rho \ll \pi : \mathbb{E}_{\boldsymbol{\theta} \sim \rho}[\mathcal{L}_P^l(f_{\boldsymbol{\theta}})] \leq \delta(\mathbb{E}_{\boldsymbol{\theta} \sim \rho}[\hat{\mathcal{L}}_{\mathcal{D}}^l(f_{\boldsymbol{\theta}})], \mathbb{KL}(\rho \parallel \pi), N, \varepsilon)) \geq 1 - \varepsilon,$$

where the loss on the true data distribution is denoted as  $\mathcal{L}_P^l(f_{\boldsymbol{\theta}}) = \mathbb{E}_{(\mathbf{x}, \mathbf{y}) \sim P}[l(f_{\boldsymbol{\theta}}(\mathbf{x}), \mathbf{y})]$  and the empirical loss on the training data is  $\hat{\mathcal{L}}_{\mathcal{D}}^l(f_{\boldsymbol{\theta}}) = \frac{1}{N} \sum_{i=1}^N l(f_{\boldsymbol{\theta}}(\mathbf{x}_i), \mathbf{y}_i)$ . In general, the bounds balance the fit to the available data (empirical risk term) and the similarity of the posterior to the prior (KL-divergence term), and the bounds hold with a probability greater than  $1 - \varepsilon$ . Various forms of the  $\delta$  bound can be found in the literature, with varying

degrees of tightness. For classification, we rely on the McAllester (McAllester, 1999b) and Catoni (Catoni, 2007) bounds, which are widely used in existing analysis.<sup>3</sup>

Since these bounds depend on the expected empirical loss, we have to iterate over the entire training data and sample from the posterior multiple times at each step in order to evaluate the bound once. This makes the given optimization problem computationally intractable for large models. Using the error function as a loss, we propose to compute an approximate upper bound by only using quantities that were already computed during the LA, i.e., the Fisher matrix  $\text{diag}(\{\mathbf{F}_l\}_{l \in [L]})$  and the model parameters  $\hat{\boldsymbol{\theta}}$ :

$$\mathbb{E}_{\boldsymbol{\theta} \sim \rho} \left[ \frac{1}{N} \sum_{i=1}^N \mathbb{1}[\arg \max_{\mathbf{y}'} p(\mathbf{y}' | \mathbf{x}_i, f_{\boldsymbol{\theta}}) \neq \mathbf{y}_i] \right] \leq \mathbb{E}_{\boldsymbol{\theta} \sim \rho} \left[ \frac{1}{N} \sum_{i=1}^N -\frac{\ln p(\mathbf{y}_i | \mathbf{x}_i, f_{\boldsymbol{\theta}})}{\ln 2} \right] \quad \text{where} \quad (3.8)$$

$$\mathbb{E}_{\boldsymbol{\theta} \sim \rho} \left[ \frac{1}{N} \sum_{i=1}^N -\frac{\ln p(\mathbf{y}_i | \mathbf{x}_i, f_{\boldsymbol{\theta}})}{\ln 2} \right] \approx \frac{-\ln p(\mathcal{D} | f_{\hat{\boldsymbol{\theta}}}) + \frac{1}{2} \sum_{l \in [L]} \tau_l \text{tr}(\mathbf{F}_l (\beta_l \mathbf{F}_l + \alpha_l \tilde{\mathbf{F}}_l)^{-1})}{N \ln 2},$$

where  $\tilde{\mathbf{F}}_l$  is from the precision matrix of the prior. Here, we first use a second-order Taylor approximation of around  $\hat{\boldsymbol{\theta}}$ . Furthermore, the expectation can be converted to a trace by using the cyclic property of the trace together with the fact that the posterior is a multivariate Gaussian. The negative data log-likelihood of the optimal parameters can be computed jointly with the Fisher matrix during the LA. We denote this approximation by  $\text{aer}(\alpha, \beta, \tau)$ . On the other hand, the KL-divergence can also be computed in closed form for our prior-posterior pair, since both distributions are multivariate Gaussians. Thus, we can plug both terms into the McAllester bound (Guedj, 2019) to obtain the objective

$$ma(\alpha, \beta, \tau) = \text{aer}(\alpha, \beta, \tau) + \sqrt{\frac{\text{kl}(\alpha, \beta, \tau) + \ln \frac{2\sqrt{N}}{\varepsilon}}{2N}} \quad (3.9)$$

where we write  $\text{kl}(\alpha, \beta, \tau) = \mathbb{KL}(\rho \| \pi)$  for the KL-divergence to emphasize the dependence on the scales. Similarly for the Catoni bound (Catoni, 2007), we obtain

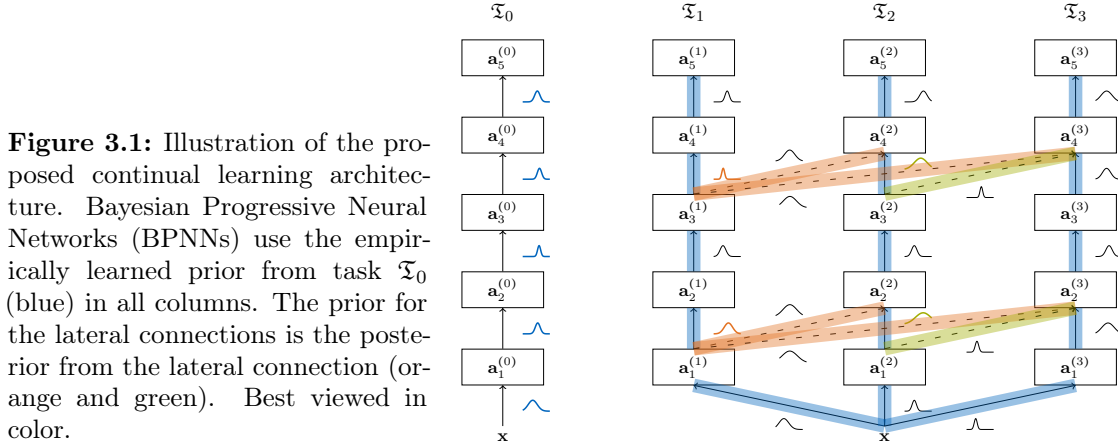
$$ca(\alpha, \beta, \tau) = \inf_{c>0} \frac{1 - \exp(-c \text{aer}(\alpha, \beta, \tau) - \frac{\text{kl}(\alpha, \beta, \tau) - \ln \varepsilon}{N})}{1 - \exp(-c)}. \quad (3.10)$$

In general, these objectives can be evaluated and minimized without using any data samples. As opposed to the cross-validated marginal likelihood or other forms of grid searching, we (a) obtain generalization bounds and analysis, (b) do not need a separate validation set, (c) can find multiple hyperparameters, i.e., scale better with the dimensionality of the considered hyperparameters due to the differentiability of the proposed objectives, and (d) the complexity of the optimization is independent of the dataset size and the complexity of the forward pass. The last point is because we only use the precomputed terms from the LA. A complete derivation of our objective is provided in A.3.

### 3.2.4. Bayesian Progressive Neural Networks

Having the essentials of learning BNN priors with generalization guarantees, we now present our extension to continual learning, which shows the versatility of the method. Here, we use the so-called Progressive Neural Networks (PNNs) (Rusu et al., 2016), where a new

<sup>3</sup>This work focuses on classification tasks as PAC-Bayes framework is usually for bounded losses. However, using recent theories of PAC-Bayes, we also comment on regression in appendix A.4.



neural network (column) is added for each new incoming task, and features from previous columns are also added for positive transfer between each task. Thus, PNNs are immune to catastrophic forgetting at the cost of an increase in memory, while being also applicable to transfer learning more generally (Wang et al., 2017; Rusu et al., 2017). As such, we extend the setup in section 3.2.2 by sequentially considering  $T$  tasks:  $\mathfrak{T}_1, \dots, \mathfrak{T}_T$ . Moreover, to keep the generality of our method, an additional task  $\mathfrak{T}_0$  is defined for the priors.

The idea behind our Bayesian reinterpretation of PNNs, dubbed BPNNs for Bayesian PNNs, is as follows. First, BNN posteriors are learned on task  $\mathfrak{T}_0$  (depicted in 3.1). Then, for an incoming sequence of tasks, the BNN priors are specified from the BNN posteriors of  $\mathfrak{T}_0$ . The proposed methods of the sums-of-Kronecker-products and PAC-Bayes bounds are used. For the lateral connections, the BNN posterior from which the lateral connection originates is used. This ensures that the weight distribution from the prior already produces reasonable features given the activations from the previous column. Together, the resulting architecture accounts for model uncertainty, generalization, and resiliency to catastrophic forgetting. All these properties are desirable to have within one unified framework. We note that BPNN is in line with our PAC-Bayes theory, that is, we increase the complexity without increasing the PAC-Bayes bounds. This is because we can reuse features from previous columns even though they do not contribute to the bounds due to their a priori fixed weight distribution. More details on BPNN are provided in appendix A.2.

### 3.3. Related Work

This chapter contributes to mainly three different areas of research. The primary area is on BNN priors, where we aim to develop a method to go beyond the isotropic Gaussian prior. In particular, we contribute to the learning-based priors for BNNs. The idea of BPNN contributes to Bayesian continual learning while we base our work on PAC-Bayes theory for improving generalization explicitly. Below, we discuss existing works in all these areas.

**Bayesian Neural Networks priors.** A prior specification is a prerequisite in Bayesian statistics. However, for neural network parameters, the choice of the prior is generally unknown. So far, uninformative priors such as isotropic Gaussian have been the de facto standard for BNNs (Fortuin, 2022). Such uninformative priors may cause undesirable effects, such as cold posteriors and worse predictive performance than standard neural networks (Wenzel et al., 2020; Fortuin et al., 2021). To this end, recent research efforts have focused on exploiting function-space (Sun et al., 2019), sparsity-inducing weight-

space (Carvalho et al., 2009; Ghosh et al., 2018), structured (Louizos & Welling, 2016), and learning-based priors (Immer et al., 2021a; Fortuin et al., 2021; Wu et al., 2019). Among these, we build upon learning-based priors. Learning-based priors can be an alternative to uninformative priors when no useful prior knowledge is available to encode, or when there exists relevant data and tasks to learn from (Fortuin, 2022).

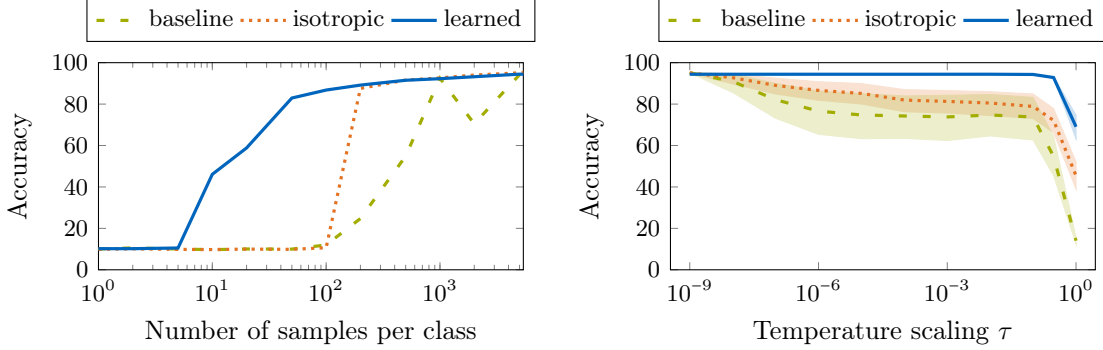
The idea of learning a prior distribution on a similar task is certainly not new. In this domain, Bayesian meta-learning (Thrun & Pratt, 1998) and their modern extensions (Rothfuss et al., 2021a; Finn et al., 2018) can be viewed as another form of learning the prior from the data. However, we note that their focus is typically not on advancing BNNs and uncertainty quantification. Empirical prior learning is closely related to our work (Robbins, 1992). For BNNs, Fortuin et al. (2021) learns the empirical weight distributions during the descent of the stochastic gradient. Wu et al. (2019) used a moment-matching approach, while Immer et al. (2021a) used the so-called Laplace-Generalized-Gauss-Newton method. In (Krishnan et al., 2020), the mean of a Gaussian prior is learned from a relevant dataset. The concurrent approaches of Shwartz-Ziv et al. (2022); Tran et al. (2022) share a similar spirit of learning expressive priors from large-scale datasets and architectures. The former utilizes the so-called SWAG (Maddox et al., 2019) framework with an inspiring idea of combining self-supervised learning, while the latter builds on Frequentist batch ensembles (Wen et al., 2020). A recent work (Rudner et al., 2023) also shows a function-space based formulation. All of these works show the strong relevance of learning-based priors for the current BNNs. Inspired by the aforementioned works, our main idea is to learn scalable and structured posteriors as expressive informative priors, like Bayesian foundation models from broader data and models. Our contributions, namely (a) the sums-of-Kronecker-product computations and (b) the PAC-Bayes bounds for LA, are another novelty of this work.

**Bayesian continual learning.** Continual learning (Thrun & Mitchell, 1995) approaches can be broadly divided into replay-based, regularization-based, and parameter-isolation methods (De Lange et al., 2022). This categorization of methods is based on the mechanisms to address catastrophic forgetting. Among these approaches, we build on parameter isolation methods, for example Rusu et al. (2016), where different model parameters are dedicated to new tasks. Within this branch, Bayesian methods have been investigated (Ebrahimi et al., 2020; Ardywibowo et al., 2022; Kumar et al., 2021), which could benefit from our work on advancing the BNN priors. Rudner et al. (2022b) shares a similar spirit of bringing the state-of-the-art BNN priors to Bayesian continual learning by relying on the function-space priors (Sun et al., 2019). On the other hand, using learning-based expressive priors, we design a single unified framework of continual learning for generalization, uncertainty awareness, and resiliency to catastrophic forgetting.

**Generalization theory.** In supervised learning, we can obtain models with a bound on the true expected loss of its true data generating process. Typical early works involved Vapnik-Chervonenkis theory and Rademacher complexity (Vapnik & Chervonenkis, 1968; Shalev-Shwartz & Ben-David, 2014). However, for high-dimensional models like neural networks, these methods often provide vacuous bounds. In recent years, the PAC-Bayes theory (McAllester, 1999b; Germain et al., 2016) has become an alternative method with wide applications. The seminar paper of (Germain et al., 2016) showed the connection to approximate Bayesian inference. Rothfuss et al. (2021b,a) devised compelling meta-learning priors for BNNs with generalization guarantees. These works form our inspiration to explore the PAC-Bayes theory for learning-based BNN priors. We note that our goal is not to advance the PAC-Bayes theory but to investigate a method for scaling the BNN priors with an approximate differentiable objective for generalization.

(a) PAC-Bayes bounds on the NotMNIST data-set using five different seeds. The ablations, namely baseline (zero mean isotropic Gaussian), grid search (GS), learned prior (LP) that is our method without PAC-Bayes, curvature scaling (CS), and Frequentist projection (FP), validate the design of our approach by each step improving the PAC-Bayes bounds, leading to a non-vacuous bound when all methods are combined.

METHOD	BASILINE	+ GS	+ LP	+ CS	+ FP
CATONI BOUND	$0.999 \pm 0.001$	$0.990 \pm 0.003$	$0.978 \pm 0.001$	$0.925 \pm 0.010$	<b><math>0.885 \pm 0.015</math></b>
MCALLESTER BOUND	$2.185 \pm 0.557$	$1.489 \pm 0.041$	$1.347 \pm 0.004$	$1.098 \pm 0.026$	<b><math>1.006 \pm 0.031</math></b>



(b) Our prior (blue, solid) needs a magnitude fewer data to have the same accuracy as an isotropic prior around zero (green, dashed) or around the pre-trained weights (orange, dotted). This means priors may help in learning from fewer data.

(c) The cold posterior effect is more prominent for the isotropic priors (zero mean: green, dashed; pre-trained mean: orange, dashed) than for the learned prior (blue, solid). This means that the temperature scale can be larger without degrading accuracy.

**Figure 3.2.:** The results of the ablation studies. The results show that the combination of our approaches can lead to non-vacuous bounds (a) and that the learned prior is more data-efficient (b) and needs less temperature scaling (c) than isotropic priors.

### 3.4. Experiments and Evaluations

The goal of the experiments is to investigate whether our approach provides generalization and calibrated uncertainty estimates. To this end, in addition to ablation studies on the presented algorithm, we show its utility for continual learning and uncertainty estimation in few-shot classification. We also closely examine the claim that our method can achieve non-vacuous bounds and also achieve state-of-the-art performance in the few-shot uncertainty benchmark. The implementation details are presented in the appendix A.5.

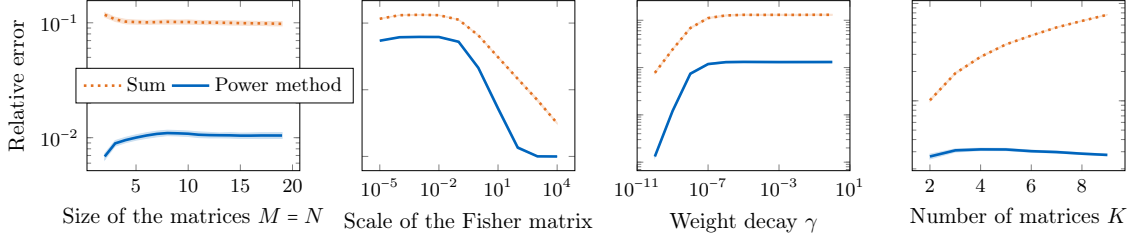
#### 3.4.1. Ablation Studies

Our method aims to bring generalization theory to the LA and also to provide an informative prior for posterior inference, which can hopefully avoid prior misspecification for better uncertainty estimates. Therefore, in the ablation studies, we want to investigate the impact of each of our methods on the generalization bounds. Moreover, we want to study the learned prior in the small-data regime and see if it can mitigate the cold posterior effect, i.e., phenomena in BNNs where the performance improves by cooling the posterior with a temperature of less than one (Wenzel et al., 2020). This effect highlights the discrepancy of current BNNs, where a posterior tempering is not theoretically needed. For this, we use a LeNet-5 architecture (Lecun et al., 1998), learn the prior on MNIST (Lecun et al., 1998) and compute the posterior on NotMNIST (Bulatov, 2011). In our experiments, we compare the performance of our learned prior against an isotropic Gaussian prior with multiple weight decays, either with a mean of zero or using the pre-trained model as the mean.



(a) Relative Frobenius error as a function of iteration. The convergence was reached in the second iteration.

ITERATION	1	2	3	4	5	6
ERROR	$0.793 \pm 0.010$	$0.049 \pm 0.001$	$0.049 \pm 0.001$	$0.049 \pm 0.001$	$0.049 \pm 0.001$	$0.049 \pm 0.001$

(b) Approximation quality on the sums-of-Kronecker products as two Kronecker factors, i.e., relative error to approximate  $\sum_{k=1}^K \mathbf{A}^k \otimes \mathbf{G}^k - \mathbf{A} \otimes \mathbf{G}$ . We vary the sizes of the matrices, the scale of the fisher matrix, the weight decay within  $(\mathbf{A} \otimes \mathbf{G} + \gamma \mathbf{I})$ , and the number of matrices for the sums-of-Kronecker-products. The results show that the proposed method can be more accurate.**Figure 3.3.:** The results of the ablation studies. The results show the accuracy of the proposed sums-of-Kronecker product computations.

In table 3.2a, one can see the contribution of each method to the final generalization bounds. For this, we report the mean and standard deviation using five different seeds. We observe that learning the prior improves the generalization bounds, as the posterior can reuse some of the pre-trained weights. The curvature further controls the flexibility of each parameter, leading to a smaller KL-divergence while still providing a small error rate. In addition, we show that scaling the curvature with our approximate bounds reduces the bounds. In particular, generalization bounds are also better than a thorough grid search. Frequentist projection further leads to a non-vacuous bound of 0.885. In the small data regime, we train our method on a random subset of NotMNIST, i.e., we vary the number of available samples per class. Figure 3.2b shows that the learned prior requires a magnitude fewer data to achieve similar accuracy compared to the isotropic priors. In appendix A.6.2, we further evaluate the impact of using different temperature scalings and weight decays in this experiment. We observe larger improvements when the model is more probabilistic, i.e., when the temperature scaling is large, and when the weight decay is small, and thus the learned prior is more dominant. Finally, following Wenzel et al. (2020), we examine the cold posterior effect using the learned prior in figure 3.2c. Although the optimal value for the temperature scaling is less than one, the temperature scaling can be closer to one compared to the isotropic priors. This suggests that the cold posterior effect is reduced by using a learned prior, which can signal that our prior is a meaningful one.

Additionally, we further validate the proposed sums-of-Kronecker product computations (see figure 3.3). In appendix A.6, further experimental results are provided, including a qualitative analysis of our tractable approximate bound used to optimize the curvature scales (appendix A.6.1). Furthermore, results for a larger cold posterior experiment using ResNet-50 (He et al., 2016), ImageNet-1K (Deng et al., 2009), and CIFAR-10 (Krizhevsky, 2009) are presented in the appendix A.6.3. Here, the learned prior improves the accuracy but does not reduce the cold posterior effect for all evaluated weight decays. This suggests the use case of our method. Overall, our results demonstrate the effectiveness of our method in improving the generalization bounds. Furthermore, we show that the learned prior is particularly useful for BNNs when there is little data available.

	OTHER	BANANA	COFFEE MUG	STAPLER	FLASHLIGHT	APPLE	AVERAGE
PNN ( $\tau = 10^{-3}$ )	95.7 $\pm$ 0.4	98.5 $\pm$ 2.3	<b>100.0 <math>\pm</math> 0.0</b>	91.7 $\pm$ 1.0	93.8 $\pm$ 9.1	93.3 $\pm$ 4.7	95.5 $\pm$ 2.9
PNN ( $\tau = 10^{-5}$ )	<b>96.7 <math>\pm</math> 0.3</b>	99.0 $\pm$ 1.2	<b>100.0 <math>\pm</math> 0.0</b>	90.7 $\pm$ 2.3	99.7 $\pm$ 0.4	94.1 $\pm$ 3.2	96.7 $\pm$ 1.2
MC DROPOUT	<u>96.3 <math>\pm</math> 0.1</u>	<b>99.3 <math>\pm</math> 0.9</b>	<b>100.0 <math>\pm</math> 0.0</b>	92.7 $\pm$ 0.8	<u>99.8 <math>\pm</math> 0.4</u>	90.4 $\pm$ 5.1	96.4 $\pm$ 1.2
BASELINE	<u>96.3 <math>\pm</math> 0.3</u>	95.5 $\pm$ 4.2	<b>100.0 <math>\pm</math> 0.1</b>	91.9 $\pm$ 1.7	<b>100.0 <math>\pm</math> 0.1</b>	87.7 $\pm$ 7.8	95.2 $\pm$ 2.4
ISOTROPIC	96.1 $\pm$ 0.3	98.7 $\pm$ 1.0	<b>100.0 <math>\pm</math> 0.0</b>	<u>93.1 <math>\pm</math> 0.6</u>	<b>100.0 <math>\pm</math> 0.0</b>	87.8 $\pm$ 6.6	96.0 $\pm$ 1.4
LEARNED	96.2 $\pm$ 0.2	98.4 $\pm$ 1.6	<b>100.0 <math>\pm</math> 0.0</b>	<b>93.9 <math>\pm</math> 0.9</b>	<b>100.0 <math>\pm</math> 0.0</b>	<b>95.1 <math>\pm</math> 4.7</b>	<b>97.3 <math>\pm</math> 1.2</b>

**Table 3.1.:** Continual learning experiment. BPNN with a learned prior (our method) improves the averaged accuracy over all tasks compared to using an isotropic prior around zero or around a learned mean. Again, we refer baseline as zero mean isotropic prior. Our method also improves over PNN with and without MC Dropout. Following Denninger & Triebel (2018), each column represents incoming continual learning tasks, where we receive new objects.

### 3.4.2. Generalization in Bayesian Continual Learning

Another advantage of our prior is its connection to continual learning. In the following experiments, we, therefore, evaluate our method for continual learning tasks. To do so, we closely follow the setup of Denninger & Triebel (2018), who introduces a practical robotics scenario. This also allows us to obtain meaningful results for practitioners. We do not use the typical MNIST setup here because we also want to test the effectiveness of the expressive prior from ImageNet (Deng et al., 2009). Each continual learning task consists of recognizing the specific object instances of the Washington RGB-D dataset (WRGBD) (Lai et al., 2011), while only a subset of all classes is available at a given time. We neglect the depth information and split the data as in Denninger & Triebel (2018). The prior is learned with ImageNet-1K (Deng et al., 2009) and ResNet-50 (He et al., 2016) is used. We specify a total of four lateral connections, each in the down-sampling  $1 \times 1$  convolution of the first “bottleneck” building block of the ResNet layers. The baselines are PNNs with several weight decays and using the MC dropout (Gal & Ghahramani, 2016). Moreover, we compare to both isotropic priors as in the ablations.

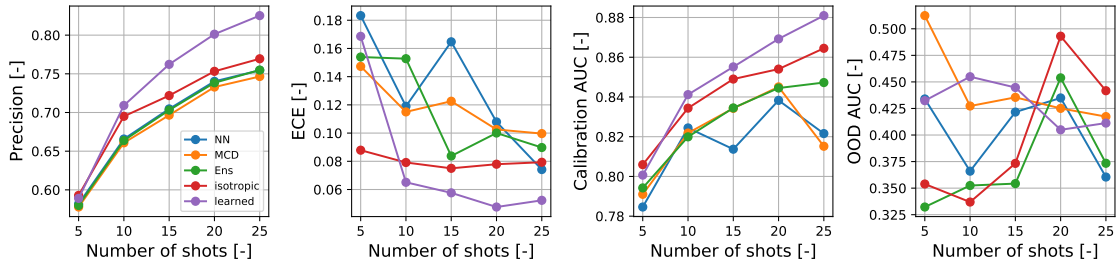
Table 3.1 shows the mean and standard deviation of the accuracy for each task of the continual learning experiment for five different random seeds. We report the mean and standard deviation of two independent runs. In our experiments, the accuracy averaged over all tasks is improved by 0.6 percent points compared to the second best approach PNN with weight decay  $10^{-5}$ . In addition, the increase in accuracy is 1.3 percent points greater compared to an isotropic prior, and 2.1 percent points greater when using zero as the prior mean. Therefore, these experimental results illustrate that with our idea of an expressive posterior as a BNN prior, we can improve the generalization on the test set for practical tasks of robotic continual learning, validating the overall design of BPNN.

### 3.4.3. Few-Shot Generalization and Uncertainty

Finally, we consider the task of few-shot learning. Our key hypothesis is that the choice of prior matters more in the small data regime i.e., the prior may dominate over the likelihood term. To this end, closely following Tran et al. (2022), we use a few-shot learning set-up across several datasets, namely CIFAR-100, UC Merced, Caltech 101, Oxford-IIIT Pets, DTD, Colorectal Histology, Caltech-UCSD Birds 200, and Cars196. CIFAR-10 is used as an out-of-distribution (OOD) dataset <sup>4</sup>. Standardized metrics such as accuracy, ECE,

<sup>4</sup>Unlike Tran et al. (2022), we did not use few shot learning on ImageNet since we obtain our prior using the entire training set.





**Figure 3.4.:** Few-shot learning experiments. The results are averaged over eight datasets. Higher is better for accuracy and AUC measures, while lower is better for ECE measures. The results show the benefits of our method in uncertainty calibration and generalization.

AUROC, and OOD AUROC are examined. The implementations are based on uncertainty baselines (Nado et al., 2021). We use ResNet-50 for the architecture.

For baselines, we choose the MC dropout (Gal & Ghahramani, 2016) (MCD) and additionally include a deep ensemble (Lakshminarayanan et al., 2017) (Ens) and a standard network (NN). These uncertainty estimation techniques are often used methods in practice (Gustafsson et al., 2020). We do not use the plex (Tran et al., 2022) due to the lack of industry-level resources to comparably train our Bayesian prior. To increase their competitiveness, we uniformly searched over ten weight decays in the range from  $10^{-1}$  to  $10^{-10}$ . We also tried fixed representation or only last-layer fine-tuning. Additionally, we add LA which represents the use of an isotropic prior. To facilitate a fair comparison between isotropic and learned prior, we carefully searched the following combinations: (a) curvature scaling without PAC-Bayes, with McAllester and Cantoni, (b) ten temperature scaling ranging from  $10^{-10}$  to  $10^{-28}$ , and (c) three weight decays  $10^{-6}$  to  $10^{-8}$  and  $10^{-10}$ . For all hyperparameters, we selected the best model using validation accuracy.

The results are reported in figure 3.4, where we averaged across eight datasets, closely following Tran et al. (2022). The results per dataset are in appendix A.6.4 whereas in appendix A.6.1, we also analyze the influence of these weight decays and temperature scales with varying numbers of available samples per class against accuracy. For shots up to 25 per class, we observe that our method often outperforms the baselines in terms of generalization and uncertainty calibration metrics. In particular, in the experiments, our learned prior from ImageNet significantly outperforms the isotropic prior within directly comparable set-ups. We interpret that the prior learning method is effective in this small data regime. This motivates the key idea of expressive posteriors as priors for BNNs. Moreover, for isotropic and learned prior, the McAllester bound often resulted in the best model. This also motivates the use of explicit curvature scaling for the generalization bounds.

We find that the use-case of our method is more within the small data regime, e.g., when the prior may dominate over the likelihood term. Secondly, one of the fundamental assumptions of prior learning methods is on the existence of relevant data and tasks to learn the prior. Moreover, for a valid PAC-Bayes analysis, the data-sets for individual tasks should not share common examples. In the future, we would like to see follow-ups that address this limitation, by either (a) learning the prior in a larger scale dataset and architectures like foundational models, or (b) combining self-supervised pretraining to quickly fine-tune the prior for the domain of the relevant task (Bommasani et al., 2021). Another direction is to obtain tighter generalization bounds by adapting clever tricks such as optimizing the entire covariance instead of individual scales with our objectives and using a subset of training data to improve the prior (Pérez-Ortiz et al., 2021).

### 3.5. Summary

This chapter presents a prior learning method for Bayesian Neural Networks. Our prior can be learned from a large-scale dataset and architecture, resulting in expressive probabilistic representations with generalization guarantees. Empirically, we demonstrated how we mitigate cold posterior effects and further obtain a non-vacuous generalization bound as low as 0.885 in neural networks using Laplace Approximation. In our benchmark experiments within continual and few-shot learning tasks, we further showed advancements over the prior arts. These results indicate that our method produces meaningful priors.

---

On Posterior Inference for Bayesian Neural Networks

---

### 4.1. Introduction

Whenever machine learning methods are used for safety-critical applications such as autonomous driving and medical image analysis, it is crucial to provide a precise estimation of the failure probability of the learned predictor. Therefore, most of the current learning approaches return distributions rather than single, most-likely predictions. However, in case of deep neural networks (DNNs), this true failure probability tends to be severely underestimated, leading to *overconfident* predictions (Guo et al., 2017). The main reason for this is that DNNs are typically trained with a principle of *maximum likelihood*, neglecting their *epistemic* or model uncertainty with the point estimates of parameters.

Imposing Gaussians on model uncertainty is arguably the most popular choice as Gaussians are for approximate inference what linear maps are for algebra. For example, once the posterior distribution is inferred, the majority of computations can be performed using the well-known tools of linear algebra. For DNNs, however, the space complexity of using Multivariate Normal Distributions (MNDs) is intractable as the covariance matrix scales quadratically with the number of parameters. Consequently, approximate inference on DNNs' posterior often neglects the parameter correlations (Wu et al., 2019; Kingma et al., 2015; Graves, 2011; Hernández-Lobato & Adams, 2015) or simplifies the covariance matrix into Kronecker products of two smaller matrices (Sun et al., 2017; Louizos & Welling, 2016; Zhang et al., 2018; Park et al., 2019) regardless of the underlying inference method.

Instead, we leverage the dual and inverse formulation of MNDs called the information form. Popularized in robotics by Thrun et al. (2004), a Gaussian is parameterized as,

$$\begin{aligned}
 p(\mathbf{x}) &\propto \exp \left\{ -\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}) \right\}, \\
 &= \exp \left\{ -\frac{1}{2}\mathbf{x}^T \boldsymbol{\Sigma}^{-1} \mathbf{x} + \boldsymbol{\mu}^T \boldsymbol{\Sigma}^{-1} \mathbf{x} \right\}, \\
 &= \exp \left\{ -\frac{1}{2}\mathbf{x}^T \mathbf{F} \mathbf{x} + \boldsymbol{\eta}^T \mathbf{x} \right\} \text{ or } \mathbf{x} \sim \mathcal{N}^{-1}(\boldsymbol{\eta}, \mathbf{F}),
 \end{aligned}$$

where  $\mathbf{x}$  is the Gaussian random variable. This formulation is fully parameterized by the information vector  $\boldsymbol{\eta}$  and matrix  $\mathbf{F}$  as opposed to the mean  $\boldsymbol{\mu}$  and covariance matrix  $\boldsymbol{\Sigma}$ .

---


$$\text{Given that } p(\mathbf{x}, \mathbf{y}) = \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_x \\ \boldsymbol{\mu}_y \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_{xx} & \boldsymbol{\Sigma}_{xy} \\ \boldsymbol{\Sigma}_{yx} & \boldsymbol{\Sigma}_{yy} \end{bmatrix}\right) = \mathcal{N}^{-1}\left(\begin{bmatrix} \boldsymbol{\eta}_x \\ \boldsymbol{\eta}_y \end{bmatrix}, \begin{bmatrix} \mathbf{F}_{xx} & \mathbf{F}_{xy} \\ \mathbf{F}_{yx} & \mathbf{F}_{yy} \end{bmatrix}\right),$$

the goal is to marginalize  $p(\mathbf{x}) = \int p(\mathbf{x}, \mathbf{y}) d\mathbf{y}$  and condition  $p(\mathbf{x}|\mathbf{y}) = \frac{p(\mathbf{x}, \mathbf{y})}{p(\mathbf{y})}$ .

---

	Marginalization	Conditioning
Covariance Form	$\boldsymbol{\mu} = \boldsymbol{\mu}_x$ $\boldsymbol{\Sigma} = \boldsymbol{\Sigma}_{xx}$	$\hat{\boldsymbol{\mu}} = \boldsymbol{\mu}_x + \boldsymbol{\Sigma}_{xy} \boldsymbol{\Sigma}_{yy}^{-1} (\mathbf{y} - \boldsymbol{\mu}_y)$ $\hat{\boldsymbol{\Sigma}} = \boldsymbol{\Sigma}_{xx} - \boldsymbol{\Sigma}_{xy} \boldsymbol{\Sigma}_{yy}^{-1} \boldsymbol{\Sigma}_{yx}$
Information Form	$\boldsymbol{\eta} = \boldsymbol{\eta}_x - \mathbf{F}_{xy} \mathbf{F}_{yy}^{-1} \boldsymbol{\eta}_y$ $\mathbf{F} = \mathbf{F}_{xx} - \mathbf{F}_{xy} \mathbf{F}_{yy}^{-1} \mathbf{F}_{yx}$	$\hat{\boldsymbol{\eta}} = \boldsymbol{\eta}_x - \mathbf{F}_{xy} \mathbf{y}$ $\hat{\mathbf{F}} = \mathbf{F}_{xx}$

---

**Table 4.1.:** Information form for a Gaussian distribution. Key differences are highlighted when compared to the widely used covariance form in terms of marginalization and conditioning. We note that conditioning is computationally simpler in information form. However, marginalization involves an inversion of a matrix. This is opposite for the covariance form.

Table 4.1 summarizes the key properties. Our major findings are that the information form has important ramifications on developing scalable *Bayesian Neural Networks*. Firstly, we point out that the approximate inference for this formulation can be simplified to scalable Laplace Approximation (LA) (MacKay, 1992c; Ritter et al., 2018b), in which we improve the state-of-the-art Kronecker-factored approximations of the information matrix (George et al., 2018) by correcting the diagonal variance in the parameter space.

More importantly, as opposed to the covariance form, the information matrix of DNNs tends to be sparse in its spectrum, i.e., eigenvalues tend to be close to zero<sup>1</sup>. Intuitively, the information content of each parameter becomes weaker with an increasing number of parameters and thus, sparsification can be effectively exploited. To do so, we propose a novel low-rank representation of the given Kronecker factorization and devise a spectral sparsification algorithm that can preserve the Kronecker product in its eigen-basis. Based on this formulation, we further demonstrate low-rank sampling computations which significantly reduce the space complexity of MND from  $O(N^3)$  to  $O(L^3)$  where  $L$  is the chosen low-rank dimension instead of the parameter space lying in high-dimensional  $N$  manifolds. Lastly, we also exhaustively perform both theoretical and empirical evaluations, yielding state-of-the-art results in both scalability and performance. In particular, we show that our method scales to large architectures trained on ImageNet, which contains over 14 million images.

**Contributions and major claims.** Our main contribution is a novel sparse representation for DNNs’ posterior that is backed up by scalable computations - more specifically: (a) an approximate inference that estimates model uncertainty in information form (section 4.2.2), (b) a low-rank representation of Kronecker-factored eigen-decomposition (section 4.2.3), (c) an algorithm to enable a low-rank approximation (LRA) for the given representation of MNDs (algorithm 2) and (d) derivation of a memory-wise tractable sampler (section 4.2.4). With our theoretical (section 4.3) and experimental results (section 4.5) we further showcase the state-of-the-art performance. Specifically, our method provably improves existing approximations. Thanks to the sparsification, our method scales to large architectures and models. Among approaches that infer structured posteriors, we achieve competitive performance while reducing the memory consumption significantly.

---

<sup>1</sup>Spectral sparsity in the 2nd order information of neural networks has been mathematically proved in (Singh et al., 2021, 2023; Singh & Hofmann, 2024) while Sagun et al. (2018) report empirical findings.

## 4.2. Posterior Inference in Sparse Information Form

### 4.2.1. Notation and Background

A neural network is a parameterized function  $f_{\boldsymbol{\theta}} : \mathbb{R}^{N_1} \rightarrow \mathbb{R}^{N_l}$  where  $\boldsymbol{\theta} \in \mathbb{R}^{N_{\boldsymbol{\theta}}}$  are the weights and  $N_{\boldsymbol{\theta}} = N_1 + \dots + N_l$ . This function  $f_{\boldsymbol{\theta}}$  is in fact a concatenation of  $l$  layers, where each layer  $i \in \{1, \dots, l\}$  computes  $\mathbf{s}_i = \mathbf{W}_i \mathbf{a}_{i-1}$  and  $\mathbf{a}_i = \sigma_i(\mathbf{s}_i)$ . Here,  $\sigma_i$  is a nonlinear function,  $\mathbf{a}_i$  are activations,  $\mathbf{s}_i$  linear pre-activations, and  $\mathbf{W}_i$  are weight matrices. The bias terms are absorbed into  $\mathbf{W}_i$  by appending 1 to each  $\mathbf{a}_i$ . Thus,  $\boldsymbol{\theta} = [\text{vec}(\mathbf{W}_1)^T \dots \text{vec}(\mathbf{W}_l)^T]^T$  where  $\text{vec}$  is the operator that stacks the columns of a matrix into a vector. Let  $\mathcal{D}\mathbf{s}_i$ , the gradient of  $\mathbf{s}_i$  w.r.t  $\boldsymbol{\theta}$ . Using LA, the posterior is approximated with a Gaussian. The mean is then given by the MAP estimate  $\hat{\boldsymbol{\theta}}$  and the covariance by the Hessian of the log-likelihood  $(\mathbf{H} + \tau \mathbf{I})^{-1}$  assuming a Gaussian prior with precision  $\tau$ . Using loss functions such as MSE or cross-entropy and piecewise linear activation  $\mathbf{a}_i$  (e.g., RELU), a good approximation of the Hessian is the Fisher Information Matrix  $\mathbf{F} = \mathbb{E}[\mathcal{D}\boldsymbol{\theta}\mathcal{D}\boldsymbol{\theta}^T]$ <sup>2</sup> for the back-propagated gradients  $\mathcal{D}\boldsymbol{\theta}$ <sup>3</sup> and is scaled by the number of data points  $N$  (Martens & Grosse, 2015). The information matrix is of size  $N_{\boldsymbol{\theta}} \times N_{\boldsymbol{\theta}}$  resulting in too large a matrix for moderately sized DNNs. Thus, computing the exact information matrix is typically intractable.

To make the computation tractable, it is first assumed that the weights across layers are uncorrelated, which corresponds to a block-diagonal form of  $\mathbf{F}$  with blocks  $\mathbf{F}_1, \dots, \mathbf{F}_l$ . Then, each realization of block  $\mathbf{F}_i$  is represented as a Kronecker product  $\mathcal{D}\boldsymbol{\theta}_i \mathcal{D}\boldsymbol{\theta}_i^T = \mathbf{a}_{i-1} \mathbf{a}_{i-1}^T \otimes \mathcal{D}\mathbf{s}_i \mathcal{D}\mathbf{s}_i^T$ . Then, matrices  $\mathbf{A}_{i-1}$  and  $\mathbf{G}_i$  are assumed to be statistically independent:

$$\mathbf{F}_i \approx \mathbf{F}_{i,\text{kfac}} = \mathbb{E}[\mathbf{a}_{i-1} \mathbf{a}_{i-1}^T] \otimes \mathbb{E}[\mathcal{D}\mathbf{s}_i \mathcal{D}\mathbf{s}_i^T] = \mathbf{A}_{i-1} \otimes \mathbf{G}_i. \quad (4.1)$$

We refer to Martens & Grosse (2015) for details on KFAC. Here,  $\mathbf{A}_{i-1} \in \mathbb{R}^{n_i \times n_i}$  and  $\mathbf{G}_i \in \mathbb{R}^{m_i \times m_i}$ , where the number of weights is  $N_i = n_i m_i$ . Typically, the information matrix is scaled by the number of data points  $N$  and incorporates the prior  $\tau$ . The herein presented parameter posterior omits the addition of prior precision and scaling term for simplicity. Here,  $N$  and  $\tau$  are treated as hyperparameters (Ritter et al., 2018b) similar to tempering in (Wenzel et al., 2020). KFAC scales to big datasets such as ImageNet (Krizhevsky et al., 2012) with large DNNs (Ba et al., 2016) and does not require changes in the training procedure when used for LA (Ritter et al., 2018b).

### 4.2.2. Approximate Inference in Information Form

We first employ an eigenvalue correction in the Kronecker-factored eigen-basis (George et al., 2018) for LA. Layer indices  $i$  are omitted, and the explanation applies layer-wise.

Let  $\mathbf{F} = \mathbf{V}_{\text{true}} \boldsymbol{\Lambda}_{\text{true}} \mathbf{V}_{\text{true}}^T$  be the true eigen-decomposition of the information matrix per layer. From this, it follows  $\boldsymbol{\Lambda}_{\text{true}} = \mathbf{V}_{\text{true}}^T \mathbf{F} \mathbf{V}_{\text{true}} = \mathbb{E}[\mathbf{V}_{\text{true}}^T \mathcal{D}\boldsymbol{\theta} \mathcal{D}\boldsymbol{\theta}^T \mathbf{V}_{\text{true}}]$  and  $\boldsymbol{\Lambda}_{\text{true},ii} = \mathbb{E}[(\mathbf{V}_{\text{true}}^T \mathcal{D}\boldsymbol{\theta})_i^2]$  where  $i \in \{1, \dots, N\}$  and  $N$  are the number of parameters of this layer. Defining the eigen-decomposition of  $\mathbf{A}$  and  $\mathbf{G}$  in equation 4.1 as  $\mathbf{A} = \mathbf{U}_A \mathbf{S}_A \mathbf{U}_A^T$  and  $\mathbf{G} = \mathbf{U}_G \mathbf{S}_G \mathbf{U}_G^T$ , it further follows  $\mathbf{F}_{\text{kfac}} \approx \mathbf{A} \otimes \mathbf{G} = (\mathbf{U}_A \otimes \mathbf{U}_G)(\mathbf{S}_A \otimes \mathbf{S}_G)(\mathbf{U}_A \otimes \mathbf{U}_G)^T$  from the properties of the Kronecker product. Now, this approximation can be improved by replacing  $(\mathbf{S}_A \otimes \mathbf{S}_G)$  with the eigenvalues  $\boldsymbol{\Lambda}_{\text{true}}$ , where  $\mathbf{V}_{\text{true}}$  is approximated with

<sup>2</sup>As we rely on LA, the Fisher naturally approximates the information matrix. However, note that in general, the information matrix does not have to be approximated with the Fisher when using other inference methods. Nevertheless, we therefore call the Fisher as information matrix in this chapter.

<sup>3</sup>The expectation is defined w.r.t the parameterized density function  $p_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x})$  assuming I.I.D. samples  $\mathbf{x}$ .

$(\mathbf{U}_A \otimes \mathbf{U}_G)$  resulting in  $\mathbf{\Lambda}_{ii} = \mathbb{E}[(\mathbf{U}_A \otimes \mathbf{U}_G)^T \mathcal{D}\boldsymbol{\theta}_i^2]$ . We denote this as the eigenvalue-corrected, Kronecker-factored eigen-basis (EFB):

$$\mathbf{F}_{\text{efb}} = (\mathbf{U}_A \otimes \mathbf{U}_G) \mathbf{\Lambda} (\mathbf{U}_A \otimes \mathbf{U}_G)^T \quad (4.2)$$

This technique has many desirable properties. Notably,  $\|\mathbf{F} - \mathbf{F}_{\text{efb}}\|_F \leq \|\mathbf{F} - \mathbf{F}_{\text{kfac}}\|_F$  w.r.t. the Frobenius norm as the computation is more accurate by correcting the eigenvalues.

However, there is an approximation in EFB since  $(\mathbf{U}_A \otimes \mathbf{U}_G)$  is still an approximation of the true eigen-basis  $\mathbf{V}_{\text{true}}$ . Intuitively, EFB only performs a correction of the diagonal elements in the eigen-basis, but when mapping back to the parameter space, this correction is again harmed by the inexact estimate of the eigenvectors. Although an exact estimation of the eigenvectors is infeasible, it is important to note that the diagonals of the exact information matrix  $\mathbf{F}_{ii} = \mathbb{E}[\mathcal{D}\boldsymbol{\theta}_i^2]$  can be computed efficiently using back-propagation. This motivates the idea to correct the approximation further as follows:

$$\mathbf{F}_{\text{inf}} = (\mathbf{U}_A \otimes \mathbf{U}_G) \mathbf{\Lambda} (\mathbf{U}_A \otimes \mathbf{U}_G)^T + \mathbf{D} \quad \text{where} \quad \mathbf{D}_{ii} = \mathbb{E}[\mathcal{D}\boldsymbol{\theta}_i^2] - \sum_{j=1}^{nm} (\mathbf{V}_{ij} \sqrt{\mathbf{\Lambda}_j})^2. \quad (4.3)$$

In equation 4.3, we have represented  $(\mathbf{U}_A \otimes \mathbf{U}_G) \mathbf{\Lambda} (\mathbf{U}_A \otimes \mathbf{U}_G)^T$  as  $\sum_{j=1}^{nm} (\mathbf{V}_{ij} \sqrt{\mathbf{\Lambda}_j})^2$  where  $\mathbf{V} = (\mathbf{U}_A \otimes \mathbf{U}_G) \in \mathbb{R}^{mn \times mn}$  is a Kronecker product with row elements  $\mathbf{V}_{ij}$  (see definition 1 below). It follows from the properties of the Kronecker product that  $i = m(\alpha - 1) + \gamma$ . The derivation is shown in appendix B.1.1. We note that the Kronecker products are never directly evaluated, but the diagonal matrix  $\mathbf{D}$  can be computed recursively, making it computationally feasible. A diagonal matrix is also efficient to store.

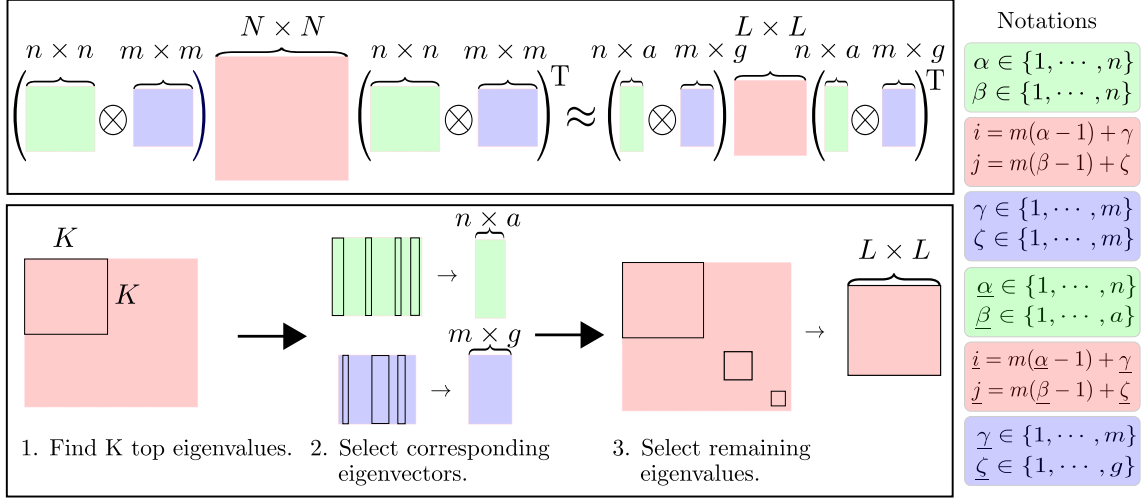
**Definition 1.** For  $\mathbf{U}_A \in \mathbb{R}^{n \times n}$  and  $\mathbf{U}_G \in \mathbb{R}^{m \times m}$ , the Kronecker product of  $\mathbf{V} = \mathbf{U}_A \otimes \mathbf{U}_G \in \mathbb{R}^{mn \times mn}$  is given by  $\mathbf{V}_{ij} = \mathbf{U}_{A_{\alpha, \beta}} \mathbf{U}_{G_{\gamma, \zeta}}$ , with  $i = m(\alpha - 1) + \gamma$  and  $j = m(\beta - 1) + \zeta$ .  $\alpha \in \{1, \dots, n\}$  and  $\beta \in \{1, \dots, n\}$  are row and column indices of  $\mathbf{U}_A$ . So as  $\gamma \in \{1, \dots, m\}$  and  $\zeta \in \{1, \dots, m\}$  for  $\mathbf{U}_G$ .

Now, the posterior distribution can be represented in an information form  $\mathcal{N}^{-1}$ :

$$\begin{aligned} p(\boldsymbol{\theta} \mid \mathcal{D}) &\sim \mathcal{N}(\hat{\boldsymbol{\theta}}, \mathbf{F}_{\text{inf}}^{-1}) \\ &= \mathcal{N}^{-1}(\hat{\boldsymbol{\theta}}_{\boldsymbol{\eta}}, (\mathbf{U}_A \otimes \mathbf{U}_G) \mathbf{\Lambda} (\mathbf{U}_A \otimes \mathbf{U}_G)^T + \mathbf{D}), \end{aligned} \quad (4.4)$$

where  $\hat{\boldsymbol{\theta}}_{\boldsymbol{\eta}}$  is an information vector of the model parameters. While we formally express the posterior as shown in equation 4.4, we do not have to explicitly compute and store the information vector. This is because, as shown later, our goal is to draw samples from the posterior. Nevertheless, this equation shows how an information form of MND can be computed using LA, and from its graphical interpretation, keeping the diagonals of the information matrix exact has also a consequence of obtaining the information content of the parameters accurately (Paskin, 2003). We note that similar insights have been studied for Bayesian tracking problems (Thrun et al., 2004) with wide adoption in practice (Bailey & Durrant-Whyte, 2006; Thrun & Liu, 2005; Eustice et al., 2006).

For predictions through marginalization with DNNs, however, the samples of the resulting posterior are to be drawn from the information matrix instead of the covariance matrix. This is because we have obtained the information matrix directly. Obviously, inverting the information naively is computationally intractable due to the large size of the matrices. To resolve these issues, an efficient sampling computation is proposed next.



**Figure 4.1.:** Illustration of algorithm 2. A low rank approximation on Kronecker-factored eigen-decomposition that preserves Kronecker structure in eigenvectors have two benefits: (a) reducing directly  $(U_A \otimes U_G)_{1:L}$  is memory-wise infeasible, and (b) sampling costs are drastically reduced as shown in section 4.2.4. Notations, low rank structure and algorithm 2 are depicted.

### 4.2.3. Model Uncertainty in Sparse Information Form

Sampling from the posterior is crucial. For example, an important use case of the posterior is estimating the predictive uncertainty for test data  $(\mathbf{x}^*, \mathbf{y}^*)$  through marginalization. This step is typically approximated with Monte Carlo integration with  $T$  samples:

$$p(\mathbf{y}^* | \mathbf{x}^*, \mathcal{D}) \approx \frac{1}{T} \sum_{t=1}^T p(\mathbf{y}^* | \mathbf{x}^*, \boldsymbol{\theta}^{(t)}) \quad \text{for } \boldsymbol{\theta}^{(t)} \sim \mathcal{N}^{-1}(\hat{\boldsymbol{\theta}}_\eta, \mathbf{F}_{\text{inf}}). \quad (4.5)$$

However, this operation is non-trivial as the sampling computation from MNDs requires  $O(N^3)$  complexity. This is due to the cost of inversion and finding a symmetrical factor from the covariance matrix. For matrices that lie in a high dimensional space, these operations are computationally infeasible. For example, if the number of parameters in a deep learning model is one million, we need to store a one million by one million matrix. Inverting such matrices and finding symmetrical factors, let alone storing them, are difficult with modern computers. While previous works [Martens & Grosse \(2015\)](#) showed that Kronecker products of two matrices can be exploited along a fidelity-cost trade-off, our main aim is to introduce an alternative form of the Gaussian posterior family.

Observing that the eigenvalues of the information matrix tend to be close to zero for the over-parameterized DNNs ([Sagun et al., 2018](#); [Singh et al., 2021, 2023](#); [Singh & Hofmann, 2024](#)), the information form can naturally leverage dimensionality reduction techniques. We note that even though the information matrix is spectrally sparse, its corresponding covariance matrix can be dense. Intuitively speaking, as more and more parameters are used to explain the same set of data, the information of these parameters tends to be smaller, and we can make use of this tendency. Indeed, in appendix B.3, we numerically show that most of the eigenvalues in an over-parameterized DNN tend to be close to zero. This means that reducing the rank of the information matrix would not induce significant error. We later show that samples can be drawn in the reduced low-dimensional space.



**Input** : Matrices  $\mathbf{U}_A, \mathbf{U}_G, \mathbf{\Lambda}$  and Rank  $K$ .  
**Output** : Matrices:  $\mathbf{U}_a, \mathbf{U}_g, \mathbf{\Lambda}_{1:L}$ .  
 1. Find top  $K$  eigenvalues  $\lambda_i$  on  $\mathbf{\Lambda}$  where  $i \in \{1, \dots, K\}$ .;  
 2. For all  $i$ , find  $\underline{\beta} = \text{floor}(\frac{i}{m}) + 1$  and  $\underline{\zeta} = i - m(\underline{\beta} - 1)$ .;  
 3. Generate sub-matrices  $\mathbf{U}_a$  and  $\mathbf{U}_g$  by selecting  
 eigenvectors in  $\mathbf{U}_A$  and  $\mathbf{U}_G$  according to  $\underline{\beta}$  and  $\underline{\zeta}$ .  
 4. Find remaining eigenvalues  $\lambda_j$  with  $j = m(\underline{\beta} - 1) + \underline{\zeta}$ .;  
 5. Concatenate and diagonalize selected eigenvalues  
 $\mathbf{\Lambda}_{1:L} = \text{diag}([\lambda_i, \lambda_j])$  for all  $i$  and  $j$ .;

**Algorithm 2:** Spectral sparsification on Kronecker-factored eigen-decomposition

Thus, we propose a low-rank form that keeps the Kronecker structure in eigenvectors <sup>4</sup>:

$$\mathbf{F}_{\text{inf}} \approx \hat{\mathbf{I}}_{\text{inf}} = (\mathbf{U}_a \otimes \mathbf{U}_g) \mathbf{\Lambda}_{1:L} (\mathbf{U}_a \otimes \mathbf{U}_g)^T + \mathbf{D}. \quad (4.6)$$

Here,  $\mathbf{U}_a$  and  $\mathbf{U}_g$  are sub-matrices of  $\mathbf{U}_A$  and  $\mathbf{U}_G$ , respectively.  $\mathbf{\Lambda}_{1:L}$  keeps top  $L$  eigenvalues from the original  $\mathbf{\Lambda}$ . Importantly, we highlight that the proposed form differs from conventional LRA, which does not preserve the Kronecker product in eigenvectors. That is, for eigenvectors, conventional LRA would use  $(\mathbf{U}_A \otimes \mathbf{U}_G)_{1:L}$  for the top  $L$  eigenvalues, and not  $(\mathbf{U}_a \otimes \mathbf{U}_g)$ . Two main advantages of this representation are that it avoids the memory-wise expensive computation of evaluating the matrix  $(\mathbf{U}_a \otimes \mathbf{U}_g)$ , where  $\mathbf{U}_a$  and  $\mathbf{U}_g$  are sub-matrices of  $\mathbf{U}_A$  and  $\mathbf{U}_G$ , respectively. This formulation also results in sampling computation that is  $O(L^3)$  in cost instead of  $O(N^3)$ , which we present later in section 4.2.4. We further note that,  $\mathbf{\Lambda}_{1:L} \in \mathbb{R}^{L \times L}$ ,  $\mathbf{U}_a \in \mathbb{R}^{m \times a}$ , and  $\mathbf{U}_g \in \mathbb{R}^{n \times g}$  denote low rank forms of corresponding eigenvalues and vectors (see figure 4.1). It follows that  $L = ag$ ,  $N = mn$ , and furthermore, the preserved rank  $L$  corresponds to preserving top  $K$  and additional  $J$  eigenvalues (resulting in  $L \geq K$ ,  $L = ag = K + J$ ) as explained with an example.

**Why can't LRA directly be used?:** Let a matrix  $\mathbf{E} = \mathbf{U}_{1:6} \mathbf{\Lambda}_{1:6} \mathbf{U}_{1:6}^T \in \mathbb{R}^{6 \times 6}$  with  $\mathbf{U}_{1:6} = [\mathbf{u}_1 \dots \mathbf{u}_6] \in \mathbb{R}^{6 \times 6}$  with  $\{\mathbf{u}_i\}_{i=1}^6$  the eigenvectors and  $\mathbf{\Lambda}_{1:6} = \text{diag}(\mathbf{\Lambda}_1, \dots, \mathbf{\Lambda}_6) \in \mathbb{R}^{6 \times 6}$  in a descending order. In this toy example, LRA with top 3 eigenvalues results:  $\mathbf{E}_{1:3} = \mathbf{U}_{1:3} \mathbf{\Lambda}_{1:3} \mathbf{U}_{1:3}^T \in \mathbb{R}^{6 \times 6}$ . Instead, if the eigenvector matrices are expressed in Kronecker product structure,  $\mathbf{E}_{\text{kron}} = (\mathbf{U}_{A_{1:3}} \otimes \mathbf{U}_{G_{1:2}}) \mathbf{\Lambda}_{1:6} (\mathbf{U}_{A_{1:3}} \otimes \mathbf{U}_{G_{1:2}})^T \in \mathbb{R}^{6 \times 6}$ . For LRA, it's not trivial to directly preserve the top 3 eigenvalues  $\mathbf{\Lambda}_{1:3}$  and corresponding eigenvectors  $(\mathbf{U}_{A_{1:3}} \otimes \mathbf{U}_{G_{1:2}})_{1:3}$ . Because as  $(\mathbf{U}_{A_{1:3}} \otimes \mathbf{U}_{G_{1:2}})_{1:3} = [\mathbf{u}_{A_1} \otimes \mathbf{u}_{G_1} \quad \mathbf{u}_{A_1} \otimes \mathbf{u}_{G_2} \quad \mathbf{u}_{A_2} \otimes \mathbf{u}_{G_1}]$ , preserving the eigenvectors with Kronecker structure results in having to store more eigenvectors:  $\mathbf{U}_{A_{1:2}} = [\mathbf{u}_{A_1} \quad \mathbf{u}_{A_2}]$  and  $\mathbf{U}_{G_{1:2}} = [\mathbf{u}_{G_1} \quad \mathbf{u}_{G_2}]$ . Consequently, additional eigenvalue  $\mathbf{\Lambda}_4$  needs to be saved so that  $\mathbf{E}_{\text{kron}_{1:3}} = (\mathbf{U}_{A_{1:2}} \otimes \mathbf{U}_{G_{1:2}}) \mathbf{\Lambda}_{1:4} (\mathbf{U}_{A_{1:2}} \otimes \mathbf{U}_{G_{1:2}})^T \in \mathbb{R}^{6 \times 6}$ .

Then, how can we develop a LRA algorithm that preserves Kronecker structures in eigenvectors? For this, we propose algorithm 2, which is also visually illustrated in figure 4.1. To select the additional eigenvectors and -values correctly, we need to introduce a definition on the indexing rules of Kronecker-factored diagonal matrices.

**Definition 2.** For diagonal matrices  $\mathbf{S}_A \in \mathbb{R}^{n \times n}$  and  $\mathbf{S}_G \in \mathbb{R}^{m \times m}$ , the Kronecker product of  $\mathbf{\Lambda} = \mathbf{S}_A \otimes \mathbf{S}_G \in \mathbb{R}^{mn \times mn}$  is given by  $\mathbf{\Lambda}_i = \mathbf{S}_{\alpha\beta} \mathbf{S}_{\gamma\zeta}$ , where the indices  $i = m(\beta - 1) + \zeta$  with  $\beta \in \{1, \dots, m\}$  and  $\zeta \in \{1, \dots, n\}$ . Then, given  $i$  and  $m$ ,  $\beta = \text{floor}(\frac{i}{m}) + 1$  and given  $\beta$ ,  $m$ , and  $i$ ,  $\zeta = i - m(\beta - 1)$ .

Notations in algorithm 2 are also depicted in figure 4.1. Now we explain this computation with a toy example below, which provides better intuition for the proposed algorithm.

<sup>4</sup>D is added after LRA which is computed similar to equation 4.3.



**Algorithm 1 for a toy example.** To explain, the same toy example can be revisited. Firstly, we aim to preserve the top 3 eigenvalues,  $i \in \{1, 2, 3\}$  which are indices of eigenvalues  $\Lambda_{1:3}$  (step 1). Then,  $\underline{\beta} \in \{1, 2\}$  and  $\underline{\zeta} \in \{1, 2\}$  can be computed using definition 2 (step 2). This relation holds as  $\Lambda$  is computed from  $\mathbf{S}_A \otimes \mathbf{S}_G$ , and thus,  $\mathbf{U}_A$  and  $\mathbf{U}_G$  are their corresponding eigenvectors respectively. Then we produce  $\mathbf{U}_{A_{1:2}}$  and  $\mathbf{U}_{G_{1:2}}$  according to  $\underline{\beta}$  and  $\underline{\zeta}$  (step 3). Again, in order to fulfill the Kronecker product operation, we need to find the eigenvalues  $j \in \{1, 2, 3, 4\}$ , and preserve  $\Lambda_{1:4}$  (step 4&5). This results in saving top 3 and additional 1 eigenvalues. Algorithm 1 provides the generalization of these steps and even if eigen-decomposition does not come with a descending order, the same logic applies.

Now, given the LRA formulation, we propose the low-rank sampling computations.

#### 4.2.4. Low-Rank Sampling Computations

Consider drawing samples  $\boldsymbol{\theta}^{(t)} \in \mathbb{R}^{mn}$  from the proposed formulation of the posteriors:

$$\boldsymbol{\theta}^{(t)} \sim \mathcal{N}^{-1}(\hat{\boldsymbol{\theta}}_{\eta}, (\mathbf{U}_a \otimes \mathbf{U}_g) \Lambda_{1:L} (\mathbf{U}_a \otimes \mathbf{U}_g)^T + \mathbf{D}). \quad (4.7)$$

Again, drawing such samples requires finding a symmetrical factor of the covariance matrix (e.g. Cholesky decomposition) which is cubic in cost  $O(N^3)$  (here  $N = mn$ ). Furthermore, in information form, it requires first an inversion of the information matrix and then the computation of a symmetrical factor which overall constitutes two operations of cost  $O(N^3)$ . Clearly, if  $N$  lies in a high dimension, sampling becomes infeasible. Therefore, we need a sampling computation that performs these operations in the dimensions of low rank  $L$ .

The proposed low-rank sampling computation is as follows. Let us define  $\mathbf{x}_0 \in \mathbb{R}^{mn}$  as the samples from a standard normal. Then, the samples  $\boldsymbol{\theta}^{(t)}$  can be computed analytically as,

$$\begin{aligned} \boldsymbol{\theta}^{(t)} &= \hat{\boldsymbol{\theta}} + \mathbf{L}^\dagger \mathbf{x}_0 \quad \text{where} \\ \mathbf{L}^\dagger &= \mathbf{D}^{-\frac{1}{2}} \left( \mathbf{I}_{nm} - \mathbf{D}^{-\frac{1}{2}} (\mathbf{U}_a \otimes \mathbf{U}_g) \Lambda_{1:L}^{\frac{1}{2}} \underbrace{(\mathbf{C}^{-1} + \mathbf{V}_s^T \mathbf{V}_s)^{-1}}_{\text{cost: } O(L^3) \ll O(N^3)} \Lambda_{1:L}^{\frac{1}{2}} (\mathbf{U}_a \otimes \mathbf{U}_g)^T \mathbf{D}^{-\frac{1}{2}} \right). \end{aligned} \quad (4.8)$$

Firstly, the symmetrical factor  $\mathbf{L}^\dagger \in \mathbb{R}^{mn \times mn}$  in equation 4.8 is a function of matrices that are feasible to store as they are diagonal or small Kronecker-factored matrices. Furthermore,

$$\mathbf{V}_s = \mathbf{D}^{-\frac{1}{2}} (\mathbf{U}_a \otimes \mathbf{U}_g) \Lambda_{1:L}^{\frac{1}{2}} \quad \text{and} \quad \mathbf{C} = \mathbf{A}_{\text{cholesky}}^{-T} (\mathbf{B}_{\text{cholesky}} - \mathbf{I}_L) \mathbf{A}_{\text{cholesky}}^{-1}, \quad (4.9)$$

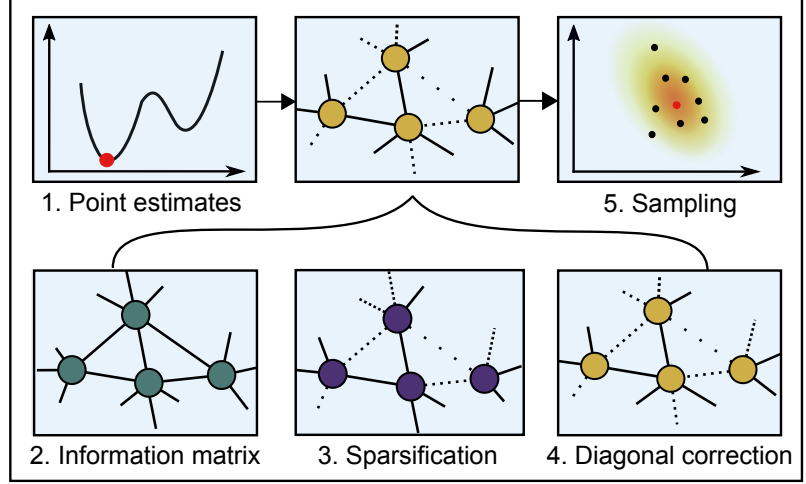
with  $\mathbf{A}_{\text{cholesky}}$  and  $\mathbf{B}_{\text{cholesky}}$  being the Cholesky decomposed matrices of  $\mathbf{V}_s^T \mathbf{V}_s \in \mathbb{R}^{L \times L}$  and  $\mathbf{V}_s^T \mathbf{V}_s + \mathbf{I}_L \in \mathbb{R}^{L \times L}$  respectively. This means, from the definition of Cholesky decomposition:

$$\mathbf{A}_{\text{cholesky}} \mathbf{A}_{\text{cholesky}}^T = \mathbf{V}_s^T \mathbf{V}_s \quad \text{and} \quad \mathbf{B}_{\text{cholesky}} \mathbf{B}_{\text{cholesky}}^T = \mathbf{V}_s^T \mathbf{V}_s + \mathbf{I}_L. \quad (4.10)$$

Consequently, the matrices in equation 4.8 are defined as  $\mathbf{C} \in \mathbb{R}^{L \times L}$ ,  $(\mathbf{C}^{-1} + \mathbf{V}_s^T \mathbf{V}_s) \in \mathbb{R}^{L \times L}$  and the identity matrix  $\mathbf{I}_L \in \mathbb{R}^{L \times L}$ . Again,  $L \ll N$  due to the spectral sparsity of the information matrix. In this way, the two operations, namely Cholesky decomposition and inversion, that are cubic in cost  $O(N^3)$  are reduced to the low-rank dimension  $L$  with complexity  $O(L^3)$ . The complete derivation is in appendix B.1.1 where we exploit the Kronecker structure in eigen-decomposition to compute  $\mathbf{L}^\dagger \mathbf{x}_0$  using the *vec* trick.

Now, how do all these concepts fit together? Figure 4.2 provides an overview.

**Figure 4.2:** The pipeline sketch. Information form is used for posterior inference. From its graphical interpretation (Paskin, 2003), the diagonal elements represent the information content of the node while its off-diagonal elements represent the link between the nodes. We make the weak links sparse while keeping the information content of the node accurate.



### 4.3. Theoretical Analysis and Guarantees

Before reporting our empirical findings, we first present analyses that provide insights behind the proposed methods. In particular, we motivate the diagonal correct term by showing provable improvements over two existing approximations. Moreover, we further examine the key properties of the proposed sparse information form for neural networks.

**More accurate information matrix.** The main theoretical results of adding a diagonal correction term to the Kronecker-factored eigen-basis are captured below.

**Lemma 4.3.1.** *Let  $\mathbf{F}$  be the real information matrix, and let  $\mathbf{F}_{inf}$  and  $\mathbf{F}_{efb}$  be the INF and EFB estimates of it, respectively. It is guaranteed to have  $\|\mathbf{F} - \mathbf{F}_{efb}\|_F \geq \|\mathbf{F} - \mathbf{F}_{inf}\|_F$ .*

*Proof.* The proof can be found in appendix B.1.  $\square$

**Corollary 4.3.2.** *Let  $\mathbf{F}_{kfac}$  and  $\mathbf{F}_{inf}$  be KFAC and our estimates of the real information matrix  $\mathbf{F}$  respectively. Then, it is guaranteed to have  $\|\mathbf{F} - \mathbf{F}_{kfac}\|_F \geq \|\mathbf{F} - \mathbf{F}_{inf}\|_F$ .*

These two results show that our formulation of the information matrix is more accurate than EFB (George et al., 2018) and KFAC (Martens & Grosse, 2015) in terms of Frobenius norm. For interested readers, find the proof  $\|\mathbf{F} - \mathbf{F}_{kfac}\|_F \geq \|\mathbf{F} - \mathbf{F}_{efb}\|_F$  in (George et al., 2018). However, we note that  $\|\mathbf{F} - \mathbf{F}_{kfac}\|_F \geq \|\mathbf{F} - \mathbf{F}_{efb}\|_F$  may not mean that  $\|\mathbf{F}^{-1} - \mathbf{F}_{kfac}^{-1}\|_F \geq \|\mathbf{F}^{-1} - \mathbf{F}_{efb}^{-1}\|_F$  or vice versa. Moreover, whether the Frobenius norm of error in the information matrix is a useful metric is yet to be seen.

**Properties of low-rank information matrix.** To the best of our knowledge, the proposed sparse information matrix, in the form of Kronecker-factored eigen-decomposition plus diagonal, has not been studied before. Therefore, we motivate its design and validity.

**Lemma 4.3.3.** *Let  $\mathbf{F}$  be the real Fisher information matrix, and let  $\hat{\mathbf{F}}_{inf}$ ,  $\mathbf{F}_{efb}$  and  $\mathbf{F}_{kfac}$  be the low-rank INF, EFB and KFAC estimates of it respectively. Then, it is guaranteed to have  $\|\text{diag}(\mathbf{F}) - \text{diag}(\mathbf{F}_{efb})\|_F \geq \|\text{diag}(\mathbf{F}) - \text{diag}(\hat{\mathbf{F}}_{inf})\|_F = 0$  and  $\|\text{diag}(\mathbf{F}) - \text{diag}(\mathbf{F}_{kfac})\|_F \geq \|\text{diag}(\mathbf{F}) - \text{diag}(\hat{\mathbf{F}}_{inf})\|_F = 0$ . Furthermore, if the eigenvalues of  $\hat{\mathbf{F}}_{inf}$  contain all non-zero eigenvalues of  $\mathbf{F}_{inf}$ , it still follows that  $\|\mathbf{F} - \mathbf{F}_{efb}\|_F \geq \|\mathbf{F} - \hat{\mathbf{F}}_{inf}\|_F$  holds.*

*Proof.* The proof can be found in appendix B.1.  $\square$

This result remarks on the optimality in capturing the diagonal variance. At the same time, our approach also becomes accurate in estimating off-diagonal entries if the information matrix contains many close-to-zero eigenvalues. Next, we check its validity.

**Lemma 4.3.4.** *The low rank matrix  $\hat{\Sigma} = ((\mathbf{U}_a \otimes \mathbf{U}_g)\mathbf{\Lambda}_{1:L}(\mathbf{U}_a \otimes \mathbf{U}_g)^T + \mathbf{D})^{-1} \in \mathbb{R}^{N \times N}$  is a non-degenerate covariance matrix if the diagonal correction matrix  $\mathbf{D}$  and LRA  $(\mathbf{U}_a \otimes \mathbf{U}_g)\mathbf{\Lambda}_{1:L}(\mathbf{U}_a \otimes \mathbf{U}_g)^T$  are both symmetric and positive definite. This condition is satisfied if  $(\mathbf{U}_a \otimes \mathbf{U}_g)\mathbf{\Lambda}_{1:L}(\mathbf{U}_a \otimes \mathbf{U}_g)^T_{ii} < \mathbb{E}[\mathcal{D}\theta_i^2]$  for all  $i \in \{1, 2, \dots, d\}$  and with  $\mathbf{\Lambda}_{1:L} \not\equiv 0$ .*

*Proof.* The proof can be found in appendix B.1.  $\square$

This comments on the validity of the resulting posterior (a sufficient condition only) and proves that sparsifying the matrix can lead to a valid non-degenerate covariance if two conditions are met. As non-degenerate covariance can have a uniquely defined inverse, it is important to check these two conditions. We note that searching the rank can be automated with off-line computations that do not involve any data. Thus, it does not introduce significant overhead. In case  $\mathbf{D}$  does not turn out to be, eigenvalue clipping (Chen et al., 2018) or finding nearest positive semi-definite matrices (Higham, 1988) can still be used. For a side note, above Lemma provides a sufficient condition, and even if  $\mathbf{D}$  is not positive definite, there is no indication that the given representation is an invalid form of covariance. These conditions have been a conservative guideline to make the likelihood term non-degenerate, which we found to work well in practice. Lastly, the inverse  $\mathbf{D}^{-1}$  is more numerically stable when we add a prior precision term and a scaling factor  $(\mathbf{N}\mathbf{D} + \tau\mathbf{I})^{-1}$ .

Before introducing the next theoretical property, let us define  $\hat{\mathbf{F}}_{1:K}^{\text{top}}$  and  $\hat{\mathbf{F}}_{1:L}$  as,

$$\hat{\mathbf{F}}_{1:K}^{\text{top}} = (\mathbf{U}_a \otimes \mathbf{U}_g)_{1:K} \mathbf{\Lambda}_{1:K} (\mathbf{U}_a \otimes \mathbf{U}_g)_{1:K}^T \text{ and } \hat{\mathbf{F}}_{1:L} = (\mathbf{U}_a \otimes \mathbf{U}_g) \mathbf{\Lambda}_{1:L} (\mathbf{U}_a \otimes \mathbf{U}_g)^T. \quad (4.11)$$

$\hat{\mathbf{F}}_{1:K}^{\text{top}}$  is a low-rank EFB estimate of the true Fisher that preserves the top K eigenvalues. Similarly,  $\hat{\mathbf{F}}_{1:L}$  is our proposal that preserves the Kronecker structure in eigenvectors. Now,

**Lemma 4.3.5.** *Let  $\mathbf{F} \in \mathbb{R}^{N \times N}$  be the real Fisher information matrix, and let  $\hat{\mathbf{F}}_{1:K}^{\text{top}} \in \mathbb{R}^{N \times N}$ ,  $\hat{\mathbf{F}}_{1:L}^{\text{top}} \in \mathbb{R}^{N \times N}$  and  $\hat{\mathbf{F}}_{1:L} \in \mathbb{R}^{N \times N}$  be the low-rank estimates of  $\mathbf{F}$  of EFB obtained by preserving top K, L and top K plus additional J resulting in L eigenvalues. Here, we define  $K < L$ . Then, the approximation error of  $\hat{\mathbf{F}}_{1:L}$  is as follows:  $\|\mathbf{F} - \hat{\mathbf{F}}_{1:L}^{\text{top}}\|_F \geq \|\mathbf{F} - \hat{\mathbf{F}}_{1:L}\|_F \geq \|\mathbf{F} - \hat{\mathbf{F}}_{1:K}^{\text{top}}\|_F$ .*

*Proof.* The proof can be found in appendix B.1.  $\square$

This bound provides an insight that if preserving top L eigenvalues results in a prohibitively large covariance matrix, our LRA provides an alternative to preserving top K eigenvalues given that  $K < L$ . In practice, note that  $\hat{\mathbf{F}}_{1:L}$  is memory-wise feasible because we can separately store  $\mathbf{U}_a$ ,  $\mathbf{U}_g$ , and  $\mathbf{\Lambda}_{1:L}$ . In contrast, evaluating  $(\mathbf{U}_a \otimes \mathbf{U}_g)$  and storing  $(\mathbf{U}_a \otimes \mathbf{U}_g)_{1:K}$  is currently not feasible for large models with millions of parameters per layer.

## 4.4. Related Work

This chapter mainly contributes to the Laplace Approximation for neural networks. To do so, we draw inspiration from sparse information filters in the robotics literature. As we also propose a new formulation of the information matrix, our contributions can also be reviewed among research works that approximate the Hessian of neural networks. Our sparsification algorithm also adds a new technique for dimensionality reduction.

**Sparse information filters.** [Thrun et al. \(2004\)](#) proposed the extended sparse information filter (SEIF), which is a dual form of the extended Kalman filter. A key property of SEIF is that all update equations can be executed in constant time, which is achieved by relying on the information form and its sparsity. Our work brings the key ideas of SEIF into the context of approximate Bayesian inference for DNNs.

**Approximation of the hessian.** The Hessian of DNNs is prohibitively large as its size is quadratic to the number of parameters. For this problem, an efficient approximation is a layer-wise Kronecker factorization ([Martens & Grosse, 2015](#); [Botev et al., 2017](#)) with demonstrably impressive scalability ([Ba et al., 2016](#)). In a recent extension by [George et al. \(2018\)](#) the eigenvalues of the Kronecker-factored matrices are re-scaled so that the diagonal variance in its eigen-basis is exact. The work demonstrates a provable method of achieving improved performance. We heavily build upon these for Bayesian DNNs, as well-built software infrastructures such as ([Dangel et al., 2020](#)) already exist.

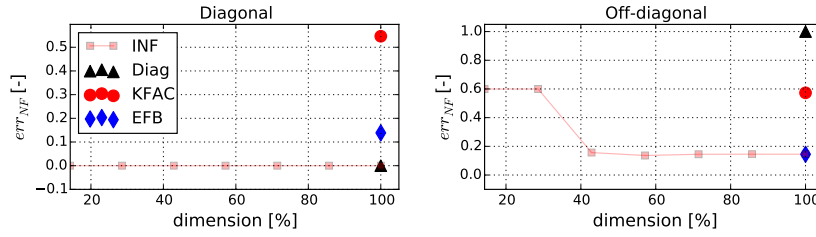
**Laplace Approximation** Instead of methods rooted in variational inference ([Hinton & van Camp, 1993](#)) and sampling ([Neal, 1996b](#)), we utilize LA ([MacKay, 1992c](#)) for the inference principle. Recently, diagonal ([Becker & Lecun, 1989](#)) and Kronecker-factored approximations ([Botev et al., 2017](#)) to the Hessian have been applied to LA by [Ritter et al. \(2018b\)](#). The authors have further proposed using LA in continual learning ([Ritter et al., 2018a](#)) and demonstrate competitive results by significantly outperforming its benchmarks ([Kirkpatrick et al., 2017](#); [Zenke et al., 2017](#)). Building upon [Ritter et al. \(2018b\)](#) for approximate inference, we propose a more expressive posterior distribution than the matrix normal distribution. Concurrently, [Kristiadi et al. \(2020\)](#) provides a formal statement on how approximate inference such as LA can mitigate the overconfident behavior of DNNs with ReLU for a binary classification case. In the context of variational inference, SLANG ([Mishkin et al., 2018](#)) shares a similar spirit to ours in using a low-rank plus diagonal form of covariance, where the authors show the benefits of low-rank approximation in detail. Yet, SLANG is different from ours as it does not explore Kronecker structures. SWA ([Maddox et al., 2018](#)), SWAG ([Maddox et al., 2019](#)) and subspace inference ([Izmailov et al., 2020](#)) have also demonstrated strong results by exploring the insights on the loss landscape of DNNs. We also acknowledge that other alternative paradigms are also promising. Some examples are post-hoc calibrations ([Guo et al., 2017](#); [Wenger et al., 2020](#)), deep ensembles ([Lakshminarayanan et al., 2017](#)) and combining Bayesian Neural Networks with probabilistic graphical models such as Conditional Random Fields ([Feng et al., 2019](#)).

**Dimensionality reduction.** A vast literature exists for dimensionality reduction beyond principal component analysis ([Wold et al., 1987](#)) and singular value decomposition ([Golub & Reinsch, 1971](#); [Van Der Maaten et al., 2009](#)). To our knowledge, though, dimensionality reduction in Kronecker-factored eigen-decomposition has not been studied before.

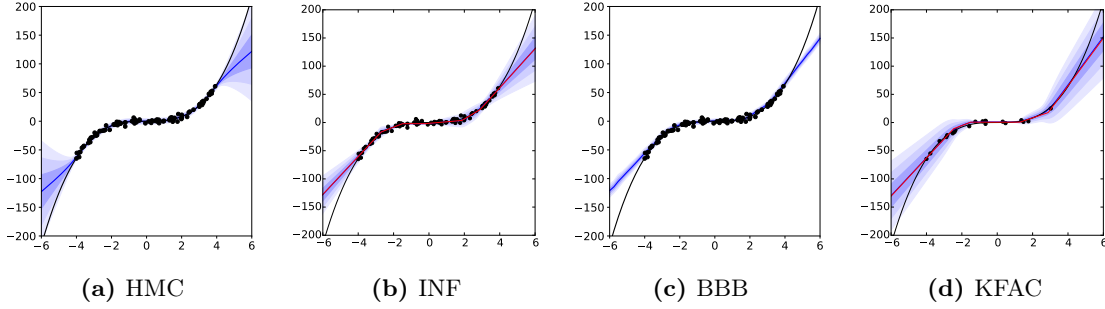
## 4.5. Experiments and Evaluations

We perform exhaustive empirical analysis across regression, classification, and active learning tasks. The chosen datasets are toy regression, UCI ([Dua & Graff, 2017](#)), MNIST ([Lecun et al., 1998](#)), CIFAR10 ([Krizhevsky, 2009](#)) and ImageNet ([Krizhevsky et al., 2012](#)) datasets. In total, ten baselines with three default LA-based methods (Diag, KFAC, and EFB) are compared. We also closely examine the effects of varying degrees of LRA.

Our main aim is to introduce the sparse information form of MND in the context of Bayesian Deep Learning, and thus, the experiments are designed to show the insight that spectral sparsification does not induce significant approximation errors while reducing



**Figure 4.3:** Effects of Low Rank Approximation in Frobenius norm of error. Lower the better. EFB, Diag, KFAC and INF are compared to exact information matrix.



**Figure 4.4.:** Evaluating predictive uncertainty in a toy regression. The black dots and the black lines are data points ( $\mathbf{x}$ ,  $\mathbf{y}$ ). The red and blue lines show predictions of the deterministic network and the mean output respectively. Upto three standard deviations are shown with blue shades.

the space complexity of using an expressive and structured posterior distribution. More importantly, we demonstrate that our method scales to datasets of ImageNet size and large architectures, and further benchmark against existing methods. We note that all the layers of the model are treated Bayesian. Implementation details are in appendix B.2.

#### 4.5.1. Small Scale Experiments

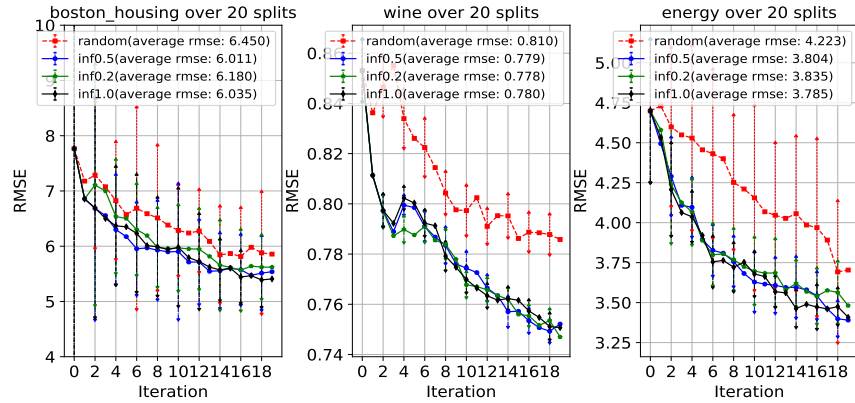
Firstly, our empirical evaluations on toy regression and UCI datasets are presented. Due to the small scale of the set-up, these experiments have advantages that we can not only evaluate the quality of uncertainty estimates, but also directly compare various approximations to the true Hessian with LRA. The latter empirically validates our theoretic claims. Furthermore, the analysis connects the qualities of uncertainty estimates and the approximations of the Hessian. For the toy regression problem, we consider a single-layered network with seven units. We have used 100 uniformly distributed points  $x \sim U(-4, 4)$  and samples  $y \sim N(x^3, 3^2)$ . On UCI datasets, we closely follow [Hernández-Lobato & Adams \(2015\)](#), in which each dataset is split into 20 sets. A single layered network with 50 units is used with the exception of the protein dataset, where we have used 100 units.

We initially perform a direct evaluation of the computed information matrix with a measure on the normalized Frobenius norm of error  $err_{NF}$  w.r.t. the block-wise exact information matrix. Note that this is only possible for the small network architectures. The results are shown in figure 4.3 and table 4.2 for toy regression and UCI datasets, respectively. Across all the experiments, we find that INF is exact on diagonal elements regardless of the ranks, while KFAC and EFB induce significant errors. On off-diagonals, INF with full rank performs similarly to EFB, while decreasing the ranks of INF tends to increase the errors. Interestingly, INF with only 5% of the ranks often outperforms KFAC by a significant margin (e.g. Naval, Power, Protein). These observations are expected by the design of our approach and highlight the benefits of the information form. As the exact diagonals of

Dataset	Diagonals				Off-diagonals			
	KFAC	EFB	INF	INF (5%)	KFAC	EFB	INF	INF (5%)
<i>Boston</i>	0.238±0.019	0.296±0.015	<b>0.000±0.000</b>	<b>0.000±0.000</b>	0.672±0.021	<b>0.524±0.006</b>	<b>0.524±0.006</b>	<i>0.608±0.009</i>
<i>Concrete</i>	0.185±0.020	0.253±0.008	<b>0.000±0.000</b>	<b>0.000±0.000</b>	0.632±0.018	<b>0.506±0.008</b>	<b>0.506±0.008</b>	0.639±0.008
<i>Energy</i>	0.138±0.035	0.335±0.029	<b>0.000±0.000</b>	<b>0.000±0.000</b>	0.646±0.012	<b>0.504±0.006</b>	<b>0.504±0.006</b>	<i>0.619±0.012</i>
<i>Kin8nm</i>	0.077±0.008	0.256±0.020	<b>0.000±0.000</b>	<b>0.000±0.000</b>	0.594±0.005	<b>0.526±0.005</b>	<b>0.526±0.005</b>	0.670±0.003
<i>Naval</i>	0.235±0.024	0.224±0.026	<b>0.000±0.000</b>	<b>0.000±0.000</b>	0.716±0.029	<b>0.465±0.003</b>	<b>0.465±0.003</b>	<i>0.480±0.003</i>
<i>Power</i>	0.113±0.012	0.252±0.011	<b>0.000±0.000</b>	<b>0.000±0.000</b>	0.681±0.006	<b>0.492±0.008</b>	<b>0.492±0.008</b>	<i>0.570±0.009</i>
<i>Protein</i>	0.323±0.067	0.332±0.043	<b>0.000±0.000</b>	<b>0.000±0.000</b>	0.779±0.040	<b>0.541±0.021</b>	<b>0.541±0.021</b>	<i>0.548±0.019</i>
<i>Wine</i>	0.221±0.021	0.287±0.022	<b>0.000±0.000</b>	<b>0.000±0.000</b>	0.638±0.006	<b>0.535±0.009</b>	<b>0.535±0.009</b>	0.685±0.006
<i>Yacht</i>	0.104±0.007	0.201±0.019	<b>0.000±0.000</b>	<b>0.000±0.000</b>	0.653±0.009	<b>0.516±0.007</b>	<b>0.516±0.007</b>	0.699±0.007

**Table 4.2.:** Evaluating the accuracy of information matrix on UCI datasets. The Frobenius norm of errors for diagonal and off-diagonal elements w.r.t the exact information matrix are depicted. Reported values are normalized. Here, INF (5%) indicates that 95% of the ranks are thrown away.

**Figure 4.5:** Active learning. Average test RMSE over 20 splits and their standard errors in the active learning experiments with Boston Housing, Wine and Energy datasets. The average RMSE along 20 iterations of each curve is reported inside the bracket.



the information matrix are known and simple to compute (as opposed to the covariance matrix), we can design methods with theoretical guarantees on the approximation quality of the information matrix. Moreover, as the spectrum of the information matrix tends to be sparse, LRA can be effectively exploited without inducing significant errors. For UCI experiments, we further study the varying effects of LRA in appendix B.3.

Visualization of predictive uncertainty is shown in figure 4.4 for the toy regression. Here, HMC (Neal et al., 2011) acts as ground truth while we compare our approach to KFAC and Bayes-by-backprop or BBB (Blundell et al., 2015). The hyperparameter sets for KFAC are chosen similar to (Ritter et al., 2018b) while INF did not require the tuning of hyperparameters (after ensuring that the information matrix is non-degenerate similar to MacKay (1992c)). All the methods show higher uncertainty in the regimes far away from the training data, with BBB showing the most difference to HMC. Furthermore, KFAC predicts rather high uncertainty even within the regions that are covered by the training data. INF produces the most comparable fit to HMC with accurate uncertainty estimates. In appendix B.3, more comparison studies can be found. We also report the results of UCI experiments where we compare the reliability of uncertainty estimates using the test log-likelihood as a measure and demonstrate competitive results as well as limitations.

#### 4.5.2. Active Learning

We next show that our method can also perform a downstream task such as active learning. In this scenario, we further study the effects of LRA. For this purpose, we choose three UCI datasets (Boston housing, Wine, and Energy) closely following Hernández-Lobato & Adams (2015). In detail, the model is a neural network with a single layer of ten hidden



Measure	NN	Diag	KFAC	MC-dropout	Ensemble	EFB	INF
Accuracy and ECE evaluated on MNIST (I.I.D). Entropy evaluated on notMNIST (OOD).							
<i>Accuracy</i>	0.993	0.9935	0.9929	0.9929	<b>0.9937</b>	0.9929	0.9927
<i>ECE</i>	0.395	0.0075	0.0078	0.0105	0.0635	0.012	<b>0.0069</b>
<i>Entropy</i>	0.055±0.133	0.555 ± 0.196	0.599 ± 0.199	0.562 ± 0.19	0.596 ± 0.133	0.618 ± 0.185	<b>0.635 ± 0.19</b>
Accuracy and ECE evaluated on CIFAR10 (I.I.D). Entropy evaluated on SHVN (OOD).							
<i>Accuracy</i>	0.8606	<b>0.8659</b>	0.8572	N/A	0.8651	0.8638	0.8646
<i>ECE</i>	0.0819	0.0358	0.0351	N/A	0.0809	0.0343	<b>0.0084</b>
<i>Entropy</i>	0.245 ± 0.215	0.4129 ± 0.197	0.408 ± 0.197	N/A	0.370 ± 0.192	0.417 ± 0.196	<b>0.4338 ± 0.18</b>

**Table 4.3.:** Results of classification experiments. Accuracy and ECE are evaluated on in-domain distribution (MNIST and CIFAR10) whereas entropy is evaluated on out-of-distribution (notMNIST and SHVN). Lower the better for ECE. Higher the better for entropy and accuracy.

units. We employ the same criterion as MacKay (1992b) which linearizes the propagation of variance of weights to the output. Besides comparing with a baseline: random selection strategy, we also compare three different numbers of ranks (20%, 50%, 100%) to verify the influence of LRA.

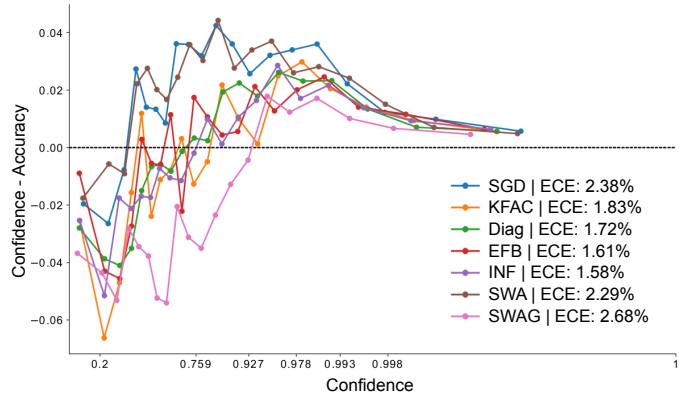
As shown in figure 4.5, uncertainty estimates of INF enable the model to learn more quickly and lead to statistically significant improvements when compared to the random selection strategy. Furthermore, even after cutting the ranks significantly, we observe that the performance can still be maintained. This can be clearly seen on both the Boston Housing and Energy datasets: the mean RMSE of the lower percentage version decreased, while their standard deviations mostly overlap. On Wine, there is nearly no decrease in performance with lower ranks. To summarize, our experiments show that INF can perform active learning, and LRA does not jeopardize the successful execution of the given task.

### 4.5.3. Classification Tasks

Next, we evaluate predictive uncertainty on classification tasks where the proposed LRA is strictly necessary. To this end, we choose the classification tasks with known and unknown classes, e.g. a network is not only trained and evaluated on MNIST but also tested on notMNIST. Note that under such tests, any probabilistic methods should report their evaluations on both known and unknown classes with the same hyperparameter settings. This is because Bayesian Neural Networks can always be highly uncertain, which may seem to work well for out-of-distribution (OOD) detection tasks but overestimates the uncertainty, even for the correctly classified samples within the training data distribution. For evaluating predictive uncertainty on known classes, Expectation Calibration Error (ECE) (Guo et al., 2017) has been used. Normalized entropy is reported for evaluating predictive uncertainty on unknown classes. LeNet with ReLU and a L2 coefficient of 1e-8 has been chosen for the MNIST dataset, which consists of two convolutional layers followed by two fully connected layers. The networks are intentionally trained to overfit or be overconfident, so that we can observe the effects of capturing model uncertainty. For CIFAR10, we choose a VGG-like architecture with two convolutional layers followed by three fully connected layers. We used batch normalization instead of dropout layers, which is a default implementation.

The results are reported in table 4.3 where we also compared to MC-dropout (Gal, 2016) and deep ensemble (Lakshminarayanan et al., 2017), which are widely used baselines in practice. For CIFAR10, we omitted MC-dropout as additionally inserting dropout layers would result in a different network and thus, the direct comparisons would not be valid. For LA-based methods, we have reported the best results after searching 300 hyperparameters

**Figure 4.6:** Reliability diagram of ResNet18 on ImageNet. Comparison of a deterministic forward pass (SGD) against Diag, KFAC, EFB, INF, SWA and SWAG are presented in terms of calibration performance. Expected Calibration Error (ECE) is reported as well. Lower the better for ECE. Closer to the center line (black), better the calibration scores in a reliability diagram.



	Diag		KFAC		EFB		INF	
Model	#Parameters	Size	#Parameters	Size	#Parameters	Size	#Parameters	Size
<b>ResNet18</b>	11,679,912	<i>44.6</i>	95,013,546	362.4	106,693,458	407.0	12,317,373	<b>47.0</b>
<b>ResNet50</b>	25,503,912	<i>97.3</i>	153,851,562	586.9	179,355,474	684.2	27,614,896	<b>105.3</b>
<b>ResNet152</b>	60,041,384	<i>229.0</i>	389,519,018	1485.9	449,560,402	1714.9	65,558,402	<b>250.1</b>
<b>DenseNet121</b>	7,895,208	<i>30.1</i>	103,094,954	393.3	110,990,162	423.4	9,711,081	<b>37.0</b>
<b>DenseNet161</b>	28,461,064	<i>108.6</i>	379,105,514	1446.2	407,566,578	1554.7	32,329,191	<b>123.3</b>

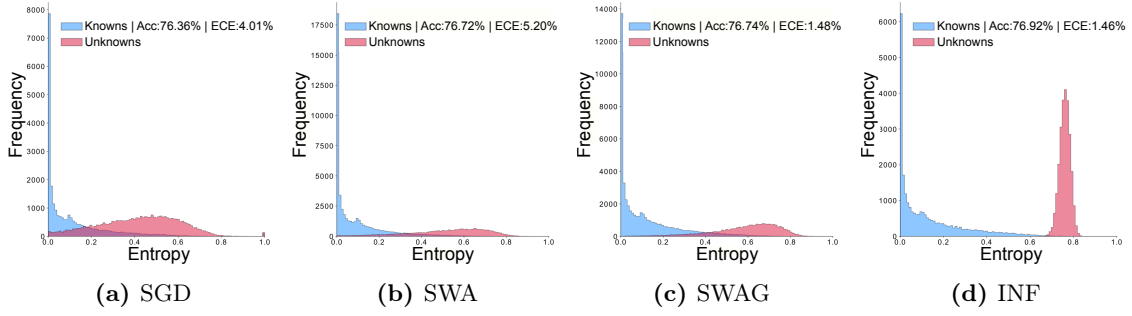
**Table 4.4.:** Network Space Complexity Comparison: The total number of information matrix parameters and its size in MB are reported for ResNet and DenseNet variants. Lower the better. Diag is only method that does not consider the correlations between the model parameters.

each. Details on the hyperparameter search are reported in appendix B.3. We find this evaluation protocol to be crucial, as LA-based methods are sensitive to these regularizing hyperparameters. Importantly, these results show that when projected to different success criteria, no inference methods largely win uniformly. Yet, these experiments also show empirical evidence that our method works in principle and compares well to the current state of the art. Estimating the posterior distribution in a sparse information form of MND, and deriving low-rank sampling computations, we show an alternative approach to designing a scalable, high-performance, and practical inference method.

#### 4.5.4. Large Scale Experiments

To show the scalability of our method, we conduct an extensive experimental evaluation on the ImageNet dataset, using five DNNs. This result alone is a key benefit of estimating model uncertainty of DNNs in sparse information form since approximate Bayesian inference only involves the computations of the information matrix in a training-free manner and the method boosts relaxed assumptions about the model when compared to MC-dropout (e.g. no specific regularizer is required). We also emphasize that the proposed diagonal correction requires only back-propagated gradients. EFB also uses the same gradients in their update step, and the whole chain takes around eight hours on ImageNet with one NVIDIA Volta GPU. This means that one can store the exact diagonals of the Fisher during EFB computations and simply add a correction term without involving any data. Thus, the added computational overhead due to the diagonal correction is negligible in practice. Similar to section 4.5.3, we evaluate both calibration and OOD detection performances. SWAG and SWA are the chosen baselines, as Maddox et al. (2019) demonstrated that these methods scale to the ImageNet dataset. Moreover, to achieve comparability for large-scale settings, we scaled the other LA-based methods up so that they are applicable





**Figure 4.7.:** Entropy histograms ResNet50: From left to right: SGD, SWA, SWAG and INF. While SGD, SWA and SWAG fail to separate in- and out-of-domain data, INF is able to almost completely differentiate between known and unknown data.

to ImageNet. This was not available in the original papers, and it constitutes an advance in evaluating LA-based methods for realistic scenarios. More importantly, we perform an extensive hyperparameter search over 100 randomly selected configurations for each LA-based method. We believe that such protocols are required for fair evaluations of LA-based methods.

To directly compare the calibration across different methods, a variant of the reliability diagram (Maddox et al., 2019) is used (figure 4.6 for ResNet 18). We present results for all other architectures in the appendices. Here, we observe that all LA variants significantly outperform SGD, SWA, and SWAG. Results of OOD detection tasks are reported in figure 4.7 where we also show the predictive uncertainty of the in-domain data for checking the under-confident behavior (Mund et al., 2015; Grimmer et al., 2013). We use artistic impressions and paintings of landscapes and objects as OOD data. Again, SGD, SWAG, and SWA are in turn significantly outperformed by INF, which clearly separates in-distribution and OOD data. These results show the competitiveness of our approach for real-world applications. However, we also find that all the LA-based methods almost identically yield strong results, clearly separating the in-distribution and OOD data.

While our method scales to ImageNet, we do not find that improvements in terms of Frobenius norm of error translate to performance in uncertainty estimation. This is the effect of regularizing hyperparameters ( $\tau$  and  $N$ ) which is a limitation of LA-based approaches. We find a counter-intuitive result that Diag LA performs similarly to ours and KFAC. It is therefore a strong alternative in practice where its complexity is superior to others (see table 4.4). Yet, ours, as we use rank 100 in the experiments, is significantly superior in space complexity when compared to EFB and KFAC while modeling the weight correlations. Therefore, we find the main use-case of INF: the real-time applications that cannot afford much more memory and require a structured form of model uncertainty, unlike Diag. Last but not least, the mathematical tools we develop and the idea of working on the inverse space of MND can also be useful for variational inference as an example.

## 4.6. Summary

This chapter introduces the sparse information form as an alternative Gaussian posterior family for which we presented novel mathematical tools such as a sparsification algorithm for the Kronecker-factored eigen-decomposition, and demonstrated how to efficiently sample from the resulting distribution. Our experiments show that our approach yields accurate estimates of the information matrix with theoretical guarantees, compares well to the

current methods for uncertainty estimation, scales to large-scale datasets while reducing space complexity, and can also be used for downstream tasks such as active learning.

---

On Predictive Distributions of Bayesian Neural Networks

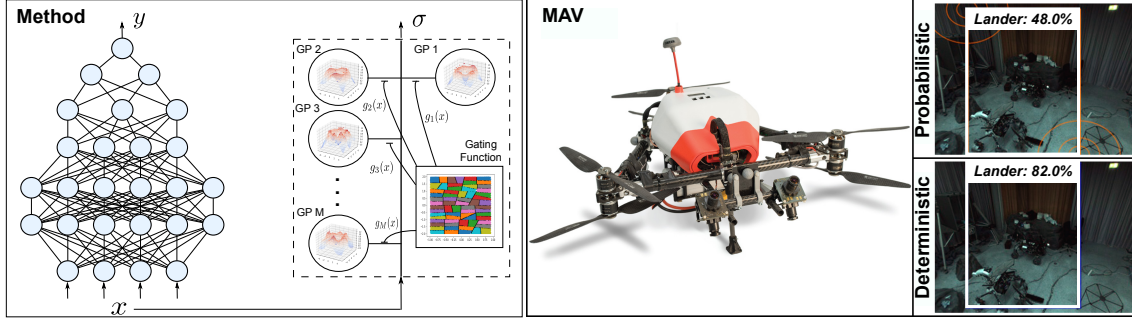
---

## 5.1. Introduction

Robots are machines that sense their environment, make decisions, and take action in the physical world. As a result, errors in their operation can lead to mission failure and, in critical applications like robotic surgery or autonomous driving, may even pose risks to human safety. This motivates probabilistic methods in robotics that pay tribute to the *uncertainty* caused by the robot’s incomplete knowledge about the world. Once equipped with an awareness of their own failures, robots will be able to avoid catastrophic effects by modifying their own behavior accordingly. Furthermore, ample evidence exists that representing information by probability distributions as opposed to relying on a single, most-likely guess can improve the reliability and robustness of robotic systems across different domains (Thrun et al., 2002; Grimmett et al., 2015, 2013).

For deep neural networks (DNNs), despite several encouraging advances (Gal & Ghahramani, 2016; Lakshminarayanan et al., 2017; Guo et al., 2017), such probabilistic methods are not ready yet for wide adoption in practice. We attribute this to the following reasons. Firstly, the current methods are often not efficient at test-time, i.e., for a single input, the methods require multiple predictions from several copies of a model (Lakshminarayanan et al., 2017), or samples from the model’s distribution (Gal & Ghahramani, 2016). Second, many existing solutions make strong assumptions about the network architecture such as the use of dropout layers (Gal & Ghahramani, 2016), or assume a readily available split of the data only for confidence calibration (Guo et al., 2017). Lastly, these methods do not generally provide reliable uncertainty estimates when compared to Gaussian Processes (GPs) - often known as the gold standard of probabilistic methods (Sun et al., 2019).

Therefore, we propose to estimate the *predictive uncertainty* of DNNs using local GPs (Jacobs et al., 1991; Tresp, 2001) - a sparse variant of GPs that divides the input space into smaller local regions using a *gating function*, where individual GPs called *experts* learn and make predictions (see figure 5.1). To do so, we provide both theoretic foundations and a practical learning algorithm. First, we formally derive a connection between DNNs and local GPs. As a result, we reveal how local GPs with a DNN-based kernel (Jacot et al., 2018) can provably approximate the uncertainty in DNNs. Moreover, we devise a learning algorithm that brings the derived theory into practice. Our solution involves a gating



**Figure 5.1.:** Left: our method computes uncertainty of neural network predictions  $\mathbf{y}$  using local GPs with the Neural Tangent Kernels (NTK). Right: with this, a Micro Aerial Vehicle (Lutz et al., 2020) performs probabilistic object detection at an interactive frame rate, reducing overconfidence of a detector. Only a feedforward network is shown but our method also applies to more broadly.

function that strictly divides the input data into smaller subsets by performing clustering in kernel space, and we propose the concept of a patchwork prior, mitigating the problem of discontinuity between local GP experts at their boundaries. For efficiency, we further propose to exploit active learning and model compression techniques.

Our approach has several favorable features for many classification and regression tasks in robotics. At run-time, it maintains the predictive power of a DNN by design, and its uncertainty estimates do not require combining multiple predictions of DNNs. Moreover, we inherit the benefits of local GPs for uncertainty estimates. These include improved scalability compared to a GP with the Neural Tangent Kernel (Jacot et al., 2018) (NTK), as well as natural support for distributed training. We contend that such probabilistic methods must run efficiently on a robot and be scalable to large datasets as much as possible. Hence, we not only provide ablation studies and evaluations against the state-of-the-art, but we also show that our method (a) scales to approximately two million data points for the task of learning inverse dynamics, and (b) on an embedded GPU of Micro Aerial Vehicles (MAV), runs more than twelve times faster than the widely used MC dropout (Gal & Ghahramani, 2016), while performing object detection for future planetary exploration.

**Contributions and major claims.** In summary, our main contribution is a novel method for estimating the predictive uncertainty of DNNs with local GPs (section 5.2.1), backed up by (a) a theoretical connection between DNNs and local GPs (section 5.2.2), (b) a divide-and-conquer solution that brings the derived theory into practice (section 5.2.3), and (c) an exhaustive empirical evaluation that demonstrates the effectiveness of our method (section 5.4). Thanks to the divide-and-conquer strategy, we claim that our method scales better than vanilla GPs by design. Our method also estimates uncertainties without sampling, which can be more efficient than sampling-based methods in terms of run time. We achieve this while improving uncertainty estimates over existing methods within public benchmarks as well as in an application scenario of planetary exploration robotics.

## 5.2. Predictive Uncertainty via Sparse Gaussian Processes

We describe a method that addresses the problem of estimating uncertainties in DNN predictions, without sampling or forming ensembles. First, we introduce the main concept of estimating predictive uncertainty with local GPs, followed by theoretic results and our learning algorithm that brings the derived theory into practical implementations.

### 5.2.1. Main Idea on Sampling-free Uncertainty Estimation

Figure 5.1 visualizes our approach. The proposed predictive model uses an existing and already well-trained DNN for accurate predictions. At the same time, the method offers the possibility to obtain the predictors’ reliable uncertainty in a closed-form solution, using DNN-based local GPs. Intuitively, we can linearize and transform DNNs around a mode to obtain a linear model. As any linear model is a GP regressor from a function-space view, we can get a GP-based representation of DNNs with a DNN-based kernel, the NTK. We describe below how the obtained GPs can estimate the uncertainty of DNNs but cannot replace the DNN predictions. Moreover, as full GPs do not scale to big data, we choose to use local GPs, which divide the data into smaller subsets and form many smaller GPs per these subsets. This results in the proposed combination of DNNs and local GPs.

**Fundamentals.** First, we introduce our notation. Considering a supervised learning task on input-output pairs  $\mathcal{D} = \{\mathcal{X}, \mathcal{Y}\} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N$ , where  $\mathbf{x}_i \in \mathbb{R}^D$ ,  $\mathbf{y}_i \in \mathbb{R}^K$ , we describe a DNN as a parametrized function  $f_{\boldsymbol{\theta}} : \mathbb{R}^D \rightarrow \mathbb{R}^K$ , where  $\boldsymbol{\theta} \in \mathbb{R}^P$ . Here, learning typically seeks to obtain an empirical risk minimizer of the loss function, i.e.,  $\min_{\boldsymbol{\theta}} \frac{1}{|\mathcal{D}|} \sum_{(\mathbf{x}, \mathbf{y}) \in \mathcal{D}} \mathcal{L}(f_{\boldsymbol{\theta}}(\mathbf{x}), \mathbf{y}) + \frac{\delta}{2} \boldsymbol{\theta}^T \boldsymbol{\theta}$  where  $\delta$  is an  $L_2$ -regularizer, and mini-batches  $\mathcal{B} \subset \mathcal{D}$  are used to find a local maximum-a-posteriori (MAP) solution  $\hat{\boldsymbol{\theta}}$ . We assume a twice differentiable and strictly convex loss function  $\mathcal{L}$ , e.g. mean squared error (MSE), and piecewise linear activations in  $f_{\boldsymbol{\theta}}$  (e.g. RELU). For a clear exposition, we drop the indices  $i$ .

To set the scene for this chapter, the Neural Linear Models (NLMs) (MacKay, 1992b; Khan et al., 2019) are defined, which can estimate a DNN’s predictive uncertainty. To do so, consider a transformed dataset  $\tilde{\mathcal{D}} = \{\mathcal{X}, \tilde{\mathcal{Y}}\}$  with the pseudo-output  $\tilde{\mathbf{y}} := \mathbf{J}_f(\mathbf{x})\hat{\boldsymbol{\theta}} - \mathbf{H}_{\mathcal{L}}(\mathbf{x}, \mathbf{y})^{-1} \mathbf{R}_{\mathcal{L}}(\mathbf{x}, \mathbf{y})$ . This transformation uses the model derivatives of the underlying DNN around  $\hat{\boldsymbol{\theta}}$ , namely the Jacobian  $\mathbf{J}_f(\mathbf{x}) := \partial f_{\boldsymbol{\theta}}(\mathbf{x}) / \partial \boldsymbol{\theta}^T \in \mathbb{R}^{K \times P}$ , the Hessian  $\mathbf{H}_{\mathcal{L}}(\mathbf{x}, \mathbf{y}) := \partial^2 \mathcal{L}(f_{\boldsymbol{\theta}}(\mathbf{x}), \mathbf{y}) / \partial f_{\boldsymbol{\theta}}(\mathbf{x})^T \partial f_{\boldsymbol{\theta}}(\mathbf{x}) \in \mathbb{R}^{K \times K}$  and the residuals  $\mathbf{R}_{\mathcal{L}}(\mathbf{x}, \mathbf{y}) := \partial \mathcal{L}(f_{\boldsymbol{\theta}}(\mathbf{x}), \mathbf{y}) / \partial f_{\boldsymbol{\theta}}(\mathbf{x}) \in \mathbb{R}^K$ . Assuming white noise and an isotropic prior, the NLMs are:

$$\tilde{\mathbf{y}} = \mathbf{J}_f(\mathbf{x})\boldsymbol{\theta} + \boldsymbol{\epsilon} \quad \text{with} \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{H}_{\mathcal{L}}(\mathbf{x}, \mathbf{y})^{-1}) \quad \text{and} \quad p(\boldsymbol{\theta}) \sim \mathcal{N}(\mathbf{0}, \delta^{-1} \mathbf{I}). \quad (5.1)$$

The NLMs can be thought of as a Bayesian linear model with learned features from a DNN, which is obtained via a linearization around the DNNs’ last layer (MacKay, 1992b). Thus, the isotropic prior is governed by the parameter  $\delta$  that corresponds to the  $L_2$  regularizer (Bishop & Nasrabadi, 2006) while the white noise is governed by the term  $\mathbf{H}_{\mathcal{L}}^{-1} \mathbf{R}_{\mathcal{L}}$  in the described data transformation, given that the DNN fits the train data well, for example,  $\mathbf{R}_{\mathcal{L}} \approx \mathbf{0}$ . The term  $\mathbf{H}_{\mathcal{L}}$  is independent of the input if the 2nd derivative of a loss is a constant, e.g. MSE. Moreover, for commonly used loss functions such as cross entropy and MSE, the covariance  $\boldsymbol{\Sigma}$  of DNN predictions  $f_{\boldsymbol{\theta}}$  is equal to that of the NLMs’  $\tilde{\mathbf{y}}$  (MacKay, 1992b; Khan et al., 2019). This means that for the same input  $\mathbf{x}$ , the NLMs can estimate the predictive uncertainty of DNNs. Yet,  $\tilde{\mathbf{y}}$  cannot be used to replace  $f_{\boldsymbol{\theta}}$ , due to their differences in  $f_{\boldsymbol{\theta}}$  and  $\tilde{\mathbf{y}}$ . In the Appendix, we provide concrete examples and their derivations.

**Main idea.** Let’s now apply a kernel trick (Rasmussen, 2003) to the NLMs, so that we can obtain an effective combination of DNNs with GPs. To explain, we can further map the NLM to a function-space, rather than working in the weight-space. In doing so, a well-known function-space formulation of linear models (Rasmussen, 2003) turns the NLM into a GP with the kernel  $\mathbf{K} = \frac{1}{\delta} \mathbf{J}_f(\mathbf{x})^T \mathbf{J}_f(\mathbf{x})$ , which is also known as the *Network Tangent Kernel* (NTK). Thus, GPs with NTK can be used to estimate the predictive uncertainty of DNNs, as their equivalent NLMs can. Here, what motivates the formulation is the idea of deep kernel machines (Wilson et al., 2016), i.e. we learn the kernel representations as opposed to

hand-designing the features and the kernels, and in our case, the kernel representations are the tangents of DNNs' Jacobian. Importantly, doing so, we get a training-free combination of DNNs with GPs, as we keep DNN predictions the same as the MAP while estimating their uncertainty using GPs with NTK.

Unfortunately, such combinations of DNNs with GPs are restricted by a prominent weakness of standard GPs (Liu et al., 2020a): cubic time complexity  $\mathcal{O}(N^3)$  that increases with the size of the dataset  $N$ . High input and output dimensions cause problems in terms of computational tractability. So, the computational costs are prohibitive for large scale problems that DNNs assume (typically referred to  $N > 10000$  in (Liu et al., 2020a; Rasmussen, 2003)). While we leave the theoretic motivation to Section 5.2.2, we thus propose local GPs with NTK, to advance the scalability of the proposed combination. A local GP consists of  $M$  experts of GPs or learners  $\mathcal{F} = \{\tilde{f}_{\text{GP}_1}, \dots, \tilde{f}_{\text{GP}_M}\}$ , and a gating function  $g: \mathbb{R}^D \rightarrow \Delta^{M-1}$  that maps any input  $\mathbf{x}$  to  $g(\mathbf{x}) = [g_1(\mathbf{x}), \dots, g_M(\mathbf{x})]$  (Jacobs et al., 1991; Tresp, 2001). These types of ensemble models are also called Mixtures of Experts (MoE) (Jacobs et al., 1991). In essence, each expert of the MoE learns and predicts within a subset of the input domain, and a gating function generates these subsets.

Specifically, we choose a gating function  $g_m(\mathbf{x}) = 1$  in just one coordinate for each input (Gross et al., 2017). This strict partition enables the use of a local model for each input (Park & Apley, 2018), avoiding the combinations of multiple predictions of a model for each input, e.g. model averaging as in Meier et al. (2014). The subscripts  $m = 1, 2, \dots, M$  denote the  $m^{\text{th}}$  expert throughout the chapter. Then, we write a local GP as,

$$\tilde{\mathbf{y}} = \sum_{m=1}^M g_m(\mathbf{x}) \tilde{f}_{\text{GP}_m}(\mathbf{x}) + \epsilon_m \quad \text{with} \quad \tilde{f}_{\text{GP}_m}(\mathbf{x}) \sim \text{GP}(\mathbf{0}, \frac{1}{\delta_m} \mathbf{J}_{f_m}(\mathbf{x})^T \mathbf{J}_{f_m}(\mathbf{x})). \quad (5.2)$$

As depicted in figure 5.1, the gating function  $g_m(\mathbf{x})$  assigns the input data  $\mathbf{x}$  to the  $m^{\text{th}}$  subset, and the individual GPs  $\tilde{f}_{\text{GP}_m}(\mathbf{x})$  learn and make predictions for the data assigned within the  $m^{\text{th}}$  subset. Consequently, the generative model and the predictions  $\mathcal{N}(\tilde{\mathbf{y}}_m^*, \Sigma_m(\mathbf{x}^*))$  on the new test datum  $\mathbf{x}^*$  are, respectively:

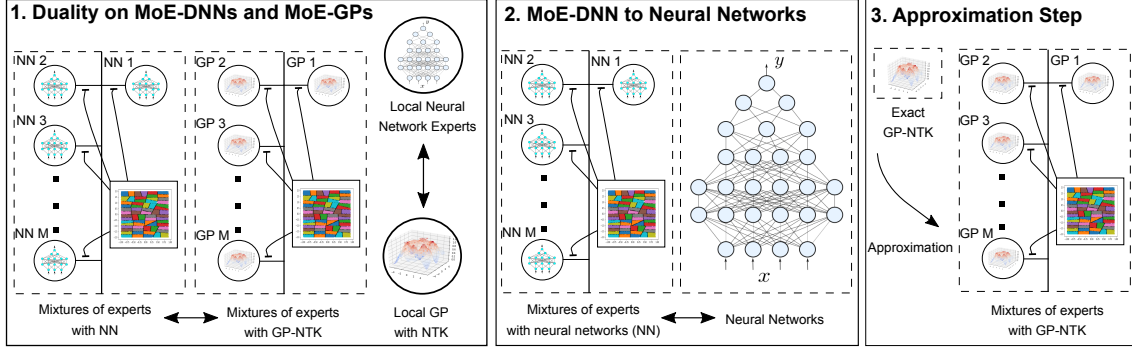
$$\begin{bmatrix} \tilde{\mathbf{y}}_m \\ \tilde{f}_{\text{GP}_m} \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K}_m + \sigma_{0,m} \mathbf{I} & \mathbf{k}_{m,*} \\ \mathbf{k}_{m,*}^T & \mathbf{k}_{m,**} \end{bmatrix}\right) \quad \text{and} \quad (5.3)$$

$$\begin{aligned} \tilde{\mathbf{y}}_m^* &= \mathbf{k}_{m,*}^T (\mathbf{K}_m + \sigma_{0,m} \mathbf{I})^{-1} \tilde{\mathbf{y}}_m, \\ \Sigma_m &= \mathbf{k}_{m,**} - \mathbf{k}_{m,*}^T (\mathbf{K}_m + \sigma_{0,m} \mathbf{I})^{-1} \mathbf{k}_{m,*} + \sigma_{0,m}, \end{aligned} \quad (5.4)$$

where,  $\mathbf{K}_m = \mathbf{K}_m(\mathcal{X}, \mathcal{X})$ ,  $\mathbf{k}_{m,*} = \mathbf{k}_m(\mathcal{X}, \mathbf{x}^*)$  and  $\mathbf{k}_{m,**} = \mathbf{k}_m(\mathbf{x}^*, \mathbf{x}^*)$ . This posterior predictive distribution  $\Sigma_m$  is an indicator of total uncertainty, i.e. the kernel function captures the model uncertainty while the constant term  $\sigma_{0,m} \mathbf{I}$  corresponds to the aleatoric uncertainty (Hüllermeier & Waegeman, 2021). In GPs, as we do not have access to exact function values, the aleatoric uncertainty often relies on the assumption of additive I.I.D. white noise, and in local GPs, the terms  $\sigma_{0,m} \mathbf{I}$  are inferred from the data per subset, leading to a non-stationary kernel. This is achieved by optimizing the marginal likelihood.

The benefits of the local GP formulation are two-fold. First, as shown above, the generative model of a GP is defined per subset, and therefore, the GP experts are smaller than the one on the entire dataset. This improves the computational complexity of GPs (Park & Apley, 2018). Second, the covariance computations are in a closed form. Thus, we neither need sampling nor ensemble (Gal & Ghahramani, 2016; Lakshminarayanan et al., 2017). Using the Lanczos approximation (Pleiss et al., 2018), which approximates matrix





**Figure 5.2.:** We visualize theoretic results as a road-map of the proof path. Detailed theorems and their proofs are in the Appendix. (1) Assuming a strict division of data, a duality between mixtures of experts (MoE) with DNNs and GPs are established first. (2) To expand the applicability of the results beyond MoE-DNN, we show that the input-prediction relationships of a single DNN and a MoE-DNN are equivalent with an assumption on the identical local DNN experts. (3) This results in a provable approximation error between the proposed local GPs and the true GPs with the NTK.

inversion with multiplications (suited for GPUs), we can compute uncertainty estimates from GPs with constant time complexity (Pleiss et al., 2018).

For classification, we leverage the framework of Lu et al. (2020), which can perform classification with the regression models via confidence calibration (Guo et al., 2017). While we refer to Lu et al. (2020) for the details, this step enables us to obtain the classification uncertainty in closed form, i.e., for a class  $c$ :

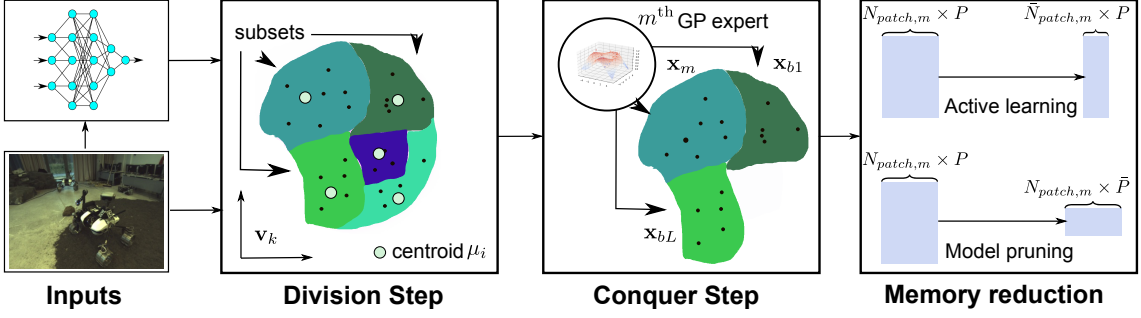
$$p(c|\mathbf{a}_m) = \text{softmax}\left(\frac{\mathbf{a}_m}{\tau_m}\right) \text{ with } \tau_m = \sqrt{1 + \lambda_{m,0}\Sigma_m(\mathbf{x}^*)}, \quad (5.5)$$

where  $\mathbf{a}_m$  is the activation,  $\tau_m$  is the temperature scaling factor, which is a function of a scaling constant  $\lambda_{m,0}$ , and GP regression uncertainty  $\Sigma_m(\mathbf{x}^*)$ . Intuitively, high prediction uncertainty will reduce the corresponding  $p(c|\mathbf{a}_m)$  by increasing the  $\tau_m$ , thus calibrating the confidences (Guo et al., 2017; Wenger et al., 2020; Lu et al., 2020).

### 5.2.2. Neural Networks as Sparse Gaussian Processes

So far, we have outlined our main idea: using the function-space view of NLMs and dividing the input space into smaller subsets, we can form smaller GPs per division. These GPs then estimate the predictive uncertainty of DNNs. Now, we discuss the theoretical results on how DNNs can be cast as local GPs with NTK. Due to space constraints, we refer the reader to the Appendix for in-depth treatment, where we provide various background materials, our theorems, and their proofs. Here, we briefly summarize the main results.

**Main result.** Figure 5.2 summarizes our findings. Consider a mixture of experts model, where the experts are DNNs and a gating function strictly divides the input space, i.e. only one local DNN expert per division (MoE-DNN). Then, instead of the estimates of the MAP parameter  $\hat{\theta}$ , we consider Bayesian DNNs with Gaussian distributions  $p(\theta|\mathcal{D})$ , i.e., representing DNNs as probability distributions over their parameters. As a first step, as a specific instance of Khan et al. (2019), we show that DNN experts have mathematically equivalent Gaussian distributions with GPs using NTK. We further show that MoE-DNNs and local GPs have equivalent Gaussian distributions, establishing the duality of the two models in a Bayesian sense. The main insight into the former is the probabilistic independence between the local experts due to a strict division of the input space.



**Figure 5.3.:** The proposed algorithm involves (a) the division of input data using Jacobian of already trained DNNs, (b) training of GP experts within their partitions and neighboring regimes, and (c) combining active learning and model pruning techniques to reduce space complexity.

Next, we establish a connection between a DNN and a local GP as shown in figure 5.2, in order to increase the applicability of our theoretical results. To do so, we point out that the input-prediction relationships of a single DNN and a MoE-DNN are equivalent if all local DNN experts are the same as the single DNN. Under the given conditions, we derive that a single DNN can be cast as a local GP, with the assumption that the data are stationary. The resulting step yields an approximation error of  $\|\mathbf{K}(\mathcal{X}, \mathcal{X}) - \mathbf{K}_{true}(\mathcal{X}, \mathcal{X})\|_F^2 = \sum_{ij} \mathbf{K}(\mathbf{x}_i, \mathbf{x}_j)^2 - \sum_{m=1}^M \sum_{ij \in \mathfrak{V}_m} \mathbf{K}(\mathbf{x}_i, \mathbf{x}_j)^2$  where  $\mathbf{K}_1 = \mathbf{K}_2 = \dots = \mathbf{K}_M$  for all  $M$  GP experts (while it is still possible to keep different hyperparameters  $\delta_m$  and  $\sigma_m$ ), and  $\mathbf{K}_{true}$  is the kernel matrix of the true DNN equivalent GP with NTK. This means that less approximation error occurs when less correlated data points are assumed to be independent by local GPs, while strongly correlated points by NTK are within the same GP experts.

### 5.2.3. A Practical Learning Algorithm

Now, we attempt to transfer our theoretical results to practice. As in figure 5.3 our solution involves the division of data into several partitions. Then, in a conquer step, we train GP experts within the partitions. Moreover, we can further reduce the memory complexity.

**Division step.** We design a gating function that divides the input data into smaller regimes s.t. highly correlated data points are grouped together whilst the points with weak correlations are separated apart. To do so, we perform NTK PCA, i.e., kernel PCA that uses NTK to reduce the dimensions of the data. Then, K-means is applied to form the clusters. Concretely, NTK PCA reduces the input data  $\mathbf{x}_i$  for  $i = 1, \dots, N$  into the low dimensional principal components  $\mathbf{v}_k = \sum_{i=1}^N \alpha_{ik} \mathbf{J}_f(\mathbf{x})^T \mathbf{J}_f(\mathbf{x}_i)$ , where  $\alpha$  is the eigenvector of  $\tilde{\mathbf{K}} \alpha_k = \lambda_k \alpha_k$  with a centered kernel matrix  $\tilde{\mathbf{K}} = \mathbf{K} - 2\mathbf{1}_N \mathbf{K} + \mathbf{1}_N \mathbf{K} \mathbf{1}_N$  for the matrix of ones  $\mathbf{1}_N \in \mathbb{R}^{N \times N}$ . Then, the centroid  $\mu_i$  and division labels  $\mathbf{c}$  are computed, iterating:

$$\mathbf{c}_i = \arg \min \sum_i^M \sum_v \|\mathbf{v}_i - \mu_i\|^2 \quad \text{and} \quad \mu_i = \frac{\sum_i^M \mathbf{1}\{\mathbf{c}_i = i\} \mathbf{v}_i}{\sum_i^M \mathbf{1}\{\mathbf{c}_i = i\}}. \quad (5.6)$$

The proposed technique gives a solution for Kernel K-means (Ding & He, 2004), which minimizes the derived error bound  $\|\mathbf{K}(\mathcal{X}, \mathcal{X}) - \mathbf{K}_{true}(\mathcal{X}, \mathcal{X})\|_F^2$  (section 5.2.2) with a balancing normalization (Dhillon et al., 2004). As kernel PCA does not scale to large datasets, we use 2-step kernel K-means (Chitta et al., 2011), which uses randomly sampled and less data to compute the cluster centroids. We refer to Chitta et al. (2011) for more details.

**Conquer step.** Next, we form the individual GP experts per divided local regime. A naive strategy is to form a single GP per regime. Unfortunately, such a model suffers from



---

```

Input : A trained DNN  $f_\theta$  and training data  $\mathcal{D} = (\mathcal{X}, \mathcal{Y})$ .
Output : Trained local GPs with a set of hyperparameters:  $\{\delta_m, \sigma_{0,m}\} \forall m$ 

begin
    Compute  $\mathbf{y} = f_\theta(\mathbf{x})$ ,  $\mathbf{J}_f(\mathbf{x})$  and  $\tilde{\mathbf{y}}$  for all training data. ;
    Apply the pruning on  $\mathbf{J}_f(\mathbf{x})$ . ; // GP compression step a (optional)
    Perform kernel PCA on NTK. ; // equation 5.6
    K-means on principal components. ; // equation 5.6
    for all the  $M$  experts do
        Apply the pruning on  $\mathbf{J}_{f_m}(\mathbf{x})$ . ; // GP compression step b (optional)
        Active learning on  $\mathbf{J}_{f_{mbL}}(\mathbf{x}) \forall l$ . ; // (optional)
        Patchwork-GP prior  $\mathbf{K}_{m,\text{patch}}$ . ; // equation 5.7
        GP expert training through marginal likelihood optimization. ;
    end
end
    
```

---

**Algorithm 3:** A divide and conquer solution (off-line).

discontinuity in predictions at the boundaries (Park & Huang, 2016). For example, input points at the cluster boundaries may yield different predictions from the surrounding GPs, causing sharp peaks in predictions. Therefore, we propose a new kernel,

$$\mathbf{K}_{m,\text{patch}} = [\mathbf{J}_{f_m}(\mathbf{x}_m), \mathbf{J}_{f_{mb1}}(\mathbf{x}_{b1}), \dots, \mathbf{J}_{f_{mbL}}(\mathbf{x}_{bL})]^T [\mathbf{J}_{f_m}(\mathbf{x}_m), \mathbf{J}_{f_{mb1}}(\mathbf{x}_{b1}), \dots, \mathbf{J}_{f_{mbL}}(\mathbf{x}_{bL})] \quad (5.7)$$

for the  $m^{\text{th}}$  GP expert. We include the neighboring GP experts  $\mathbf{J}_{f_{mb1}}, \dots, \mathbf{J}_{f_{mbL}}$  into the prior of the  $m^{\text{th}}$  expert. With this, we can incorporate the information of neighboring GPs into the prior of the  $m^{\text{th}}$  GP expert. Intuitively, the discontinuities occur at the shared boundary regimes of two GP experts. Our GP prior removes such boundary regimes locally by taking into account the neighboring regimes, mitigating the sharp peaks.

**Active uncertainty learning.** The complexity of each GP expert is bounded to  $O(N_{\text{patch},m}^3)$ , where the number of data points  $N_{\text{patch},m}$  includes the data points of neighboring GPs:  $N_{\text{patch},m} = N_m + N_{b1} + \dots + N_{bk}$ , resulting in added costs  $N_m < N_{\text{patch},m}$ . To mitigate, we perform active learning on the neighboring GP experts, and thus choose fewer, but the most informative points, e.g. IVMs (Lawrence et al., 2002; Triebel et al., 2016). Intuitively, as the neighboring data are only included for reducing the discontinuity problem, we may select fewer data points. Concretely, we use the following steps: (a) for the neighboring GPs of the  $m^{\text{th}}$  expert, we draw an initial subset and train the GPs. (b) Using the trained GP models, we rank the remaining data (stored as a pool) by uncertainty (queries generated by uncertainty sampling). (c) The GP expert is then updated with the most uncertain point. These steps repeat until  $N_{bk}$  is reduced to a desired, smaller value. In this way, while forming a patchwork prior, the neighboring GPs actively choose the neighboring data they want to learn from. As a result, we obtain:  $\bar{N}_{\text{patch},m} < N_{\text{patch},m}$ .

**Gaussian process compression.** We further reduce the complexity of the algorithm by exploiting model compression techniques. Note that the Jacobian  $\mathbf{J}_f(\mathbf{x})$  are  $N_{\text{patch},m}$  by  $P$  matrices where  $P$  is the total number of parameters in DNNs. As  $\mathbf{J}_f(\mathbf{x})$  represents the sensitivity of each parameter to the output, the elements of  $\mathbf{J}_f(\mathbf{x})$  that belong to an unimportant DNN parameter can be removed. To do so, we propose a two-staged compression technique: (a) rank the DNN parameters by their importance using existing pruning methods and remove the corresponding elements of  $\mathbf{J}_f(\mathbf{x})$  for all  $m$ . (b) Per GP

<b>Input</b> : A trained DNN $f_{\theta}$ , trained local GPs and test data $\mathbf{x}^*$ . <b>Output</b> : Prediction $\mathbf{y}^*$ and covariance $\Sigma_m$ . <b>begin</b> Compute the prediction: $\mathbf{y}^* = f_{\theta}(\mathbf{x}^*)$ ;                      // standard forward pass Compute the test Jacobian: $\mathbf{J}_f(\mathbf{x}^*)$ ; Gating network assigns $m^{th}$ expert ;                      // equation 5.6 Compute GP uncertainty $\Sigma_m$ ;                      // equation 5.4 <b>end</b>
--

**Algorithm 4:** Sampling-free predictions through marginalization (on-line).

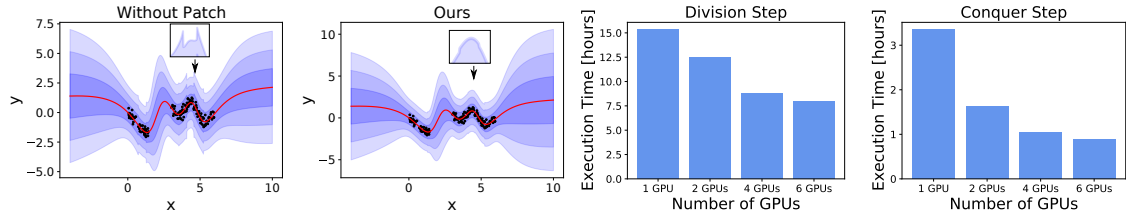
expert, rank the elements of  $\mathbf{J}_{f_m}(\mathbf{x})$  by a metric  $|\sum \mathbf{J}_{f_m}(\mathbf{x})|$ , as smaller absolute values contribute less to the kernel. The first step is targeted at removing redundant parameters in DNNs, while the second step is targeted at each individual expert, resulting in  $\bar{P} < P$ .

In algorithm 3, we summarize how local GPs can be trained. Our method is post-hoc, meaning we do not need to train a DNN together with GPs. Rather, we use an already well-trained model using existing deep learning pipelines. While algorithm 3 is an off-line procedure, algorithm 4 is what we deploy on the robots. We emphasize again that NLM (and its function-space view like local GPs) predicts pseudo output  $\tilde{\mathbf{y}}$ . This pseudo output differs from DNN predictions  $\mathbf{y}^*$  but their covariances are equal. So, we obtain predictions using DNN and estimate its uncertainty using local GPs. More importantly, our method is sampling-free because GP’s covariance can be obtained in closed-form analytic expressions.

### 5.3. Related Work

Many researchers explored the idea of robotic introspection (Grimmett et al., 2015), i.e. robots that reason about “when they don’t know” in order to avoid catastrophic failures. So far, many proposed methods have used robots’ past failure experiences (Daftary et al., 2016; Ramanagopal et al., 2018; Gurău et al., 2018), or detected inconsistencies within the predictors (Richter & Roy, 2017; Wong et al., 2020). These works provide strong evidence that introspection improves the reliability of robots. Other methods are based on uncertainty estimation (Thrun et al., 2002), where learning algorithms such as DNNs use probability theory to express their own reliability Feng et al. (2019); Harakeh et al. (2020); Sünderhauf et al. (2019). Commonly, these works use a technique called MC-dropout (Gal & Ghahramani, 2016). However, the intense computational burden in estimating uncertainty via sampling or ensembles (Gawlikowski et al., 2023) limits their applicability to real-time systems (Lutz et al., 2020; Lee et al., 2020a, 2018b).

Several techniques without sampling or ensembles have been presented for run-time efficiency. These methods often use Dirichlet distributions or functional Bayesian Neural Networks (Sensoy et al., 2018; Carvalho et al., 2020). Unfortunately, the applicability of Dirichlet distributions is limited to classification, while functional Bayesian Neural Networks face hurdles in scaling to large datasets without approximations (Carvalho et al., 2020). We recommend the recent work of He et al. (2020) and Adlam et al. (2021), who use NTK-based GPs for uncertainty estimation of DNNs and provide useful insights. However, we focus on a technique that can be readily deployed on a real-time system such as a MAV. Hence, we take a different path to ensembles of infinite width DNNs (He et al., 2020; Adlam et al., 2021). We find uncertainty propagation techniques (MacKay, 1992b; Postels et al., 2019;



**Figure 5.4.:** Left: visualizing predictive uncertainty of DNNs with and without the proposed patchwork prior. The black dots are train data points, and the red line shows the predictions of a deterministic network. Blue shades show up to three standard deviations. Without the patch prior, sharp peaks are observed in uncertainty estimates. Right: training time for approximately 2 million data points. We learned 512 GP experts per joint torques with distributed training.

(Foong et al., 2019) to be practical, as these methods are training-free or post-hoc, i.e., the method can work with existing and already well-trained DNNs.

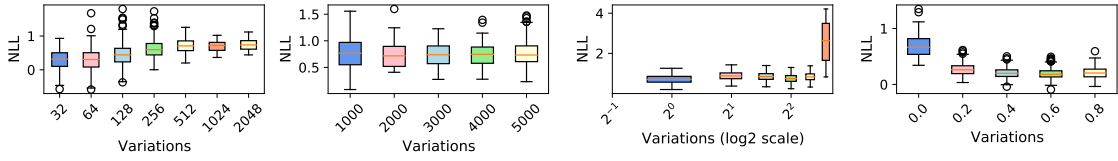
In addition, our theoretic contribution (section 5.2.2) extends on how DNNs are related to GPs (pioneered by Neal (1996a) and others (Lee et al., 2018a; Jacot et al., 2018; Khan et al., 2019)), where we derive a relationship between DNNs and local GPs as opposed to a single GP. Theoretic insights that connect DNNs and local GPs may pave the way towards their application, as GPs alone do not scale to big data required by DNNs. Lastly, we extend previous work on local GPs (Section 5.2.3). Local GPs are efficient but suffer from non-smooth uncertainty estimates (Park & Huang, 2016). Park & Apley (2018) show an elegant solution called patchwork kriging. By augmenting data at the boundaries between GP experts, experts are forced to produce similar predictions at their boundaries. However, since patchwork kriging is limited to low-dimensional inputs (Park & Apley, 2018), we propose several techniques that extend it to higher dimensions, such as images.

## 5.4. Experiments and Evaluations

Next, we provide exhaustive empirical evaluations. The goal is not only to validate the proposed theory, but also to show the promises of the proposed formulation: (a) run-time efficient sampling-free uncertainty, (b) mitigating peaked uncertainties, (c) distributed training and (d) reliable uncertainty estimates. Therefore, in addition to ablation studies, we examine metrics such as the quality of uncertainty estimates, improved scalability when compared to vanilla GPs, and run-time in comparison to sampling-based methods. Implementation details and additional results can be found in appendix C.3.

### 5.4.1. Toy Regression

With a toy regression in the Snelson dataset, we show (a) how local GPs can estimate the predictive uncertainty of DNN, and (b) the ability of the patchwork before producing smoother uncertainty estimates. For this, we train a single hidden layer Multi-layer perceptron (MLP) with 200 units and a tanh activation. Evaluating for out-of-distribution (OOD) data (points far from train data), and domain-shift (DS) data (removed in-between points), as shown in figure 5.4, the uncertainty estimates produced are high as test points move away from the train data. Moreover, when the DNN predictions match the train data, the uncertainty estimates produced are close to the data noise. We also observe that the patchwork prior mitigates the discontinuity problem of local GPs.



**Figure 5.5.:** The effects of each hyperparameter is shown with normalized test NLL, by varying them in different steps (Variations), and fixing the others to default settings. Lower the better.

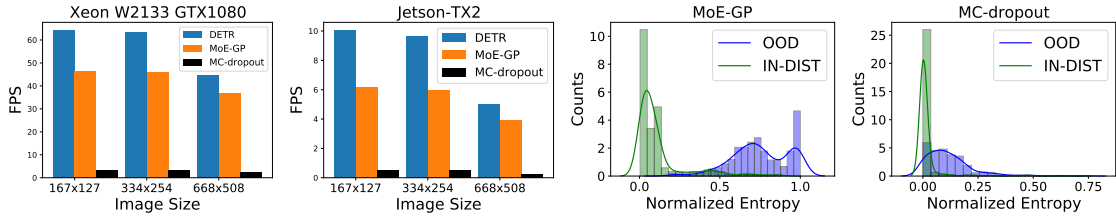
Train	Test	Ensemble	MC-dropout	rMC-dropout	Laplace	rLaplace	Ours
sarcos	sarcos	$1.261 \pm 0.105$	$1.398 \pm 0.012$	$1.398 \pm 0.012$	$1.392 \pm 0.014$	$1.392 \pm 0.014$	<b><math>1.035 \pm 0.039</math></b>
sarcos	kuka1	$17.86 \pm 2.995$	<b><math>16.21 \pm 0.866</math></b>	$20.58 \pm 1.144$	$25.88 \pm 1.354$	$24.55 \pm 2.746$	$21.53 \pm 2.268$
sarcos	kuka2	$17.50 \pm 2.895$	<b><math>15.63 \pm 0.858</math></b>	$20.17 \pm 0.988$	$25.59 \pm 1.079$	$24.42 \pm 2.257$	$20.92 \pm 2.236$
sarcos	kukasim	<b><math>23.06 \pm 2.649</math></b>	$51.09 \pm 11.14$	$61.40 \pm 12.921$	$77.13 \pm 15.40$	$74.03 \pm 13.23$	$68.95 \pm 4.003$
kuka1	kuka1	$2.013 \pm 0.020$	$1.611 \pm 0.007$	$1.620 \pm 0.006$	$1.676 \pm 0.013$	$1.719 \pm 0.071$	<b><math>1.347 \pm 0.013</math></b>
kuka1	kuka2	$2.085 \pm 0.005$	$1.349 \pm 0.019$	$1.330 \pm 0.006$	<b><math>1.310 \pm 0.005</math></b>	<b><math>1.310 \pm 0.008</math></b>	$1.315 \pm 0.003$
kuka1	kukasim	$60.44 \pm 1.108$	$42.14 \pm 2.869$	$48.76 \pm 0.566$	$60.37 \pm 0.799$	$59.74 \pm 1.571$	<b><math>1.348 \pm 0.012</math></b>
kuka1	sarcos	$8.128 \pm 0.169$	$22.24 \pm 4.288$	$42.36 \pm 2.398$	$122.68 \pm 13.32$	$1837 \pm 2431.3$	<b><math>1.356 \pm 0.014</math></b>
kuka2	kuka2	$2.042 \pm 0.009$	$1.443 \pm 0.005$	$1.423 \pm 0.006$	$1.429 \pm 0.006$	$1.423 \pm 0.006$	<b><math>1.354 \pm 0.013</math></b>
kuka2	kukasim	$63.07 \pm 0.741$	$38.94 \pm 0.561$	$48.93 \pm 0.779$	$60.36 \pm 1.005$	$59.05 \pm 1.158$	<b><math>1.333 \pm 0.012</math></b>
kuka2	sarcos	$8.509 \pm 0.461$	$18.24 \pm 1.599$	$53.24 \pm 6.287$	$141.7 \pm 17.14$	$211.3 \pm 180.2$	<b><math>1.355 \pm 0.014</math></b>
kuka2	kuka1	$2.106 \pm 0.010$	$1.461 \pm 0.004$	$1.395 \pm 0.004$	$1.373 \pm 0.005$	$1.374 \pm 0.004$	<b><math>1.331 \pm 0.013</math></b>

**Table 5.1.:** The results of inverse dynamic experiments are reported using NLL. Lower the better.

#### 5.4.2. Learning Inverse Dynamics

**On hyperparameters.** Next, we examine (a) the influences of hyperparameter choices and (b) provide a simple tuning recipe. For this, we examine a regression task using a five-layered 200-unit MLP on the SARCOS robot arm dataset [Rasmussen \(2003\)](#). Here, we vary each hyperparameter while fixing the remaining ones, and we compute the Negative Log Likelihood (NLL). The results are shown in figure 5.5, with four hyperparameters, namely, (a) the number of GP experts, (b) the subset sizes for clustering, (c) the pruning level, and (d) the size of the informative data points. We observe the following: (a) the NLL is proportional to the number of GP experts, (b) the subset size does not significantly influence the NLL, (c) pruning steps have a tipping point where the NLL increases, and (d) the active learning parameter also has a tipping point, where the NLL decrease is marginal. In contrast, by design, the computational complexity increases with (a) a lower number of experts and (b) the size of actively selected points, and decreases with the pruning steps. We find that these results confirm our intuition. For example, keeping 2048 GP experts for SARCOS results in 21 data points per expert on average. This setting is inferior to keeping 32 GP experts, which have 1,390 data points on average. Reflecting this, our strategy is to assign more data points to each GP expert while selecting only the active learning points needed. The pruning levels can then be varied within a range that does not deteriorate the performance to reduce the computational complexity.

**Comparison study.** Now, we evaluate the performance of our method using the datasets namely SARCOS, KUKA1, and KUKA2 ([Meier et al., 2014](#)). The baselines are MC-dropout [Gal & Ghahramani \(2016\)](#), Laplace Approximation ([Ritter et al., 2018b](#); [Lee et al., 2020b](#)), and their fast variants: rMC-dropout ([Postels et al., 2019](#)) and rLaplace ([MacKay, 1992b](#); [Foong et al., 2019](#)) (see section 2.3.3). The fast variants are our main competitors: Bayesian approaches that are training-free, i.e. compatible with pre-trained DNNs without modifications to the training procedures. We find decoupling DNN training



**Figure 5.6.:** Left: run-time analysis shows efficiency of our approach when compared to MC-dropout (Gal & Ghahramani, 2016) with 20 samples. Any difference from a deterministic DNN can also be seen. Right: entropy histograms show novelty detection performance.

from uncertainty estimates crucial, as it ensures comparisons of uncertainty estimates only by keeping the accuracy of the predictors constant among the baselines (in this case, the same pretrained five-layered MLP across the baselines). We also include the ensembles (Lakshminarayanan et al., 2017). Lastly, we adopt zero-shot cross-dataset transfer to systematically evaluate in-domain, OOD, and DS scenarios, following recent insights; that is, we train on a dataset and evaluate uncertainty estimates for all other datasets.

The results are in table 5.1, where we averaged over three random initializations and report NLL as a metric (a standard for regression tasks). We find that our method often outperforms other baselines significantly. MC-dropout dominates in three experiments. Yet, MC-dropout requires sampling for uncertainty estimates - one of the core problems addressed in our work. We note again that all the compared methods are training-free, making the comparisons meaningful. Importantly, rLaplace uses the NLM from section 5.2 to infer Gaussian uncertainty in weight-space, in contrast to function-space inference with GPs. Thus, the comparison of rLaplace and ours instantiates the comparison of inference between the weight-space and function-space views. These results show the relevance of GP-based uncertainty estimates for DNNs over uncertainty propagation methods.

**On scalability.** Here, we show that (a) our approach scales to the KUKA SIM dataset (which contains 1984950 data points) and (b) supports distributed training. Scalability is relevant, as DNNs operate in the regime of big data, and an exact GP does not scale to such settings. Moreover, a key benefit of local GP over other local GPs lies in its distributed, local nature, and we can exploit it in practice. The results are depicted in figure 5.4, where we plot the training time in hours against the number of used GPUs. Using 512 GP experts and 100 iterations for marginal log likelihood optimization, we find that our method scales to approximately two million data points within a day, and less than eight hours when using six GPUs. This experiment shows empirical evidence that our method scales to two million data points and validates the claim that local GPs support distributed training.

### 5.4.3. Probabilistic Object Detection

To validate the applicability of our method to a robot and evaluate the run-time on the hardware, we perform experiments on a MAV within the context of an on-going space demo mission for future planetary exploration (Schuster et al., 2020). For this, we train an object detector, DETR (Carion et al., 2020b) with an EfficientNet backbone. For testing, we create two scenarios: (a) a carry test, used for training and testing with 1008 manual labels, and (b) flight tests with different configurations of the objects (e.g., space rover and lander). The latter creates OOD samples, with a slight domain shift in data distribution. In field robotics, the assumption that the test set comes from the same training data distribution is routinely violated due to the changes in the environments, and we attempt to create such

effects. Moreover, we also evaluate the complexity of our method on a Jetson-TX2, which is used on our MAV. The quantitative results are shown in figure 5.6, which shows that (a) our method can be fast on a Jetson-TX2 and (b) can separate OOD samples within this scenario. The accompanying video explains this setup and the qualitative results <sup>1</sup>.

If the computational complexity of GPs can be tamed in practice, our work shows that the predictive uncertainty of DNNs can be obtained from the GP formulation. By advancing the scalability of a full NTK, the advantages of our approach are demonstrated. Yet, the applicability to larger data regimes, e.g. ImageNet, is confined to that of the sparse GPs, e.g., the non-parametric models requiring access to the training data at run-time, which can require more memory than the parametric models such as DNNs. Sparse GPs may also face struggles when the output dimension is larger. Alternatives can be weight-space methods (see section 2.4). Here, various approximations have been devised so far in order to cope with the high-dimensional weight-space. For robotics though, when the availability of data is limited, the dimensions of the data space can be smaller than the weight-space, and thus, our work can provide a reference that the Bayesian non-parametric can be a relevant tool in addressing the current challenges of uncertainty estimation for neural networks.

### 5.5. Summary

This chapter presented a new approach for estimating the uncertainty of neural networks using local Gaussian Processes with the Neural Tangent Kernel. Importantly, we have established a theoretic connection between the two models, and suggested a way to exploit such duality in practice. Experimentally, we validate and demonstrate the benefits of our approach. In particular, we find that our method often leads to superior performance when compared to the state-of-the-art alternatives. Furthermore, our method can deliver both fast and accurate uncertainty estimates in real-world settings. In future work, we plan to apply the proposed methodology in field robotics settings, and study how the uncertainty estimates can be used to improve the robustness of the robotic systems.

---

<sup>1</sup>Link: <https://www.youtube.com/watch?v=vu2TnDEqDRk>

## Part III.

# Extensions and Applications to Robotics

In comparison to other approaches, Bayesian methods offer the promise (a) to incorporate domain knowledge through priors, (b) to capture model uncertainty through posterior inference, and (c) to enhance predictions with uncertainty estimates through marginalization. In the following chapters, we examine these promises by exploiting our methodological contributions in robotics applications. Firstly, we show that meaningful priors are highly suitable for their applications in stream-based active learning because incorporating domain knowledge can improve generalization under small data. Similarly, as our posterior information form can be scalably applied to large architectures, we can exploit pool-based active learning for visual deep learning models. Here, we show that model uncertainty can be creatively used beyond the selection of the most informative data. Lastly, we also discover that efficient predictions with uncertainty estimates can be effectively exploited for decision making under uncertainty within the context of shared autonomy. These results illustrate that our methods can improve the robustness of deep learning in robotics while addressing the issue on the lack of annotated training data in practical applications.





---

Stream-based Active Learning for Robust Semantic Scene Analysis

---

### 6.1. Motivation

With deep neural networks (DNNs), the performance of computer vision increased dramatically, achieving impressive results in the semantic perception tasks, such as object classification, detection, and segmentation (Kirillov et al., 2023; Oquab et al., 2024). However, such advancements in computer vision may not directly translate to the robotic semantic perception. This is because, in contrast to standard computer vision benchmarks, robots are situated in the physical world, where unpredictable events routinely occur and affect the robustness of the robot’s understanding of its own environments. An example is distributional shift scenarios, where DNNs often make unexpected errors due to test conditions being underrepresented in the training data (Ancha et al., 2024; Feng et al., 2023). Thus, to achieve robust semantic perception, several probabilistic techniques have been investigated so far, so that with uncertainty estimates, robots can reason when to trust the predictions from DNNs and when not (Sirohi et al., 2023; Lee et al., 2021; Ren et al., 2023; Lakshminarayanan et al., 2017; Gal & Ghahramani, 2016).

In this chapter, we build upon such probabilistic techniques and propose a system called CLEVER (robotiC stream-based active LEArner Via postERiors as priors). The main idea behind CLEVER is to not only obtain uncertainty estimates from DNNs for probabilistic predictions, but to further reduce the model’s uncertainty by asking support from humans and adapting the model online. We achieve this query and adaptation through so-called stream-based active learning (AL) – an autonomous learning paradigm that involves continuously selecting and labeling new data as they arrive in a stream, allowing for adaptation of the model online to changes in data distribution (Cacciarelli & Kulahci, 2024). The outcome of CLEVER is an adaptable DNN for semantic perception. In the work, we present the design of CLEVER in detail. These include a continuously adaptable DNN architecture for semantic perception, a Bayesian learning algorithm that utilizes posterior as informative priors, and a stream-based AL system that combines temporal information.

CLEVER meets several desiderata of stream-based AL with DNNs in practice. By learning priors, CLEVER is designed to generalize and estimate well-calibrated uncertainty, even with limited data availability. Such prediction capabilities are crucial to request support from humans (“query”) only when necessary. CLEVER also addresses the issue



**Figure 6.1.:** Imagine a robot encountering an unseen object, e.g., trained for apples, but tested on a T-shirt. This problem of distributional shifts induce learning algorithms to fail. We propose an active learner with Deep Neural Networks (DNNs). With our system, the robot queries a human whenever the model is uncertain, and adapt itself to reduce that uncertainty. Such capabilities can improve the robustness of DNN-based semantic perception by coping with distribution shifts.

of catastrophic forgetting, i.e., the tendency of DNNs to abruptly forget about previously learned tasks when continuously learning a new task (Wang et al., 2024). Furthermore, CLEVER can learn a new task in one minute by updating only the relevant parameters of DNNs online while using only fewer but most informative and diverse training data. In the experiments, we provide several ablation studies and comparative assessments to motivate our design choices. Finally, through a user validation study with 13 participants and the deployment of CLEVER on the humanoid robot ARMAR-6 (Asfour et al., 2019), we demonstrate the enhanced robustness in semantic perception with robots.

**Contributions and major claims.** To the best of our knowledge, CLEVER is the first stream-based active learning system with DNNs, shown in a physical system for robotic perception tasks. Moreover, unlike existing works, we apply stream-based active learning for securing robustness in semantic perception tasks. To enable this novel capability, we identify new system requirements and challenges (section 6.3), followed by CLEVER’s design that meets these requirements within a single framework (section 6.4). CLEVER is evaluated in response to these requirements. In particular, we show that our Bayesian formulation with learning-based priors enhances the practicality of CLEVER (section 6.5.1). Through a user validation study that involves arbitrary objects (section 6.5.2), and demonstrations on a humanoid robot for deformable object perception (section 6.5.3), we create distributional shift scenarios for evaluation. Even under these challenging scenarios, we show that CLEVER can eventually accomplish the given perception tasks, improving the robustness of the DNN-based semantic perception in the real world.<sup>1</sup>

## 6.2. Related Work

Our primary contribution is in the area of active learning from data stream. For this, we bring Bayesian methods for neural networks as well as interactions with humans for robot learning. Thus, we locate our work within these three research areas. Meanwhile, table 6.1 summarizes the main novelty of the proposed system w.r.t the state-of-the-art.

**Stream-based active learning.** Active Learning (AL) is a paradigm in which a learning

<sup>1</sup>Link to the project website: <https://sites.google.com/view/thecleversystem>.

	Addresses catastrophic forgetting in DNNs	Both uncertainty and generalization	Update DNNs fast, e.g., less than 1 minute	Ability to ask help and select samples
Continual learning <sup>a</sup>	✓	✗	✓	✗
Existing stream-based AL <sup>b</sup>	✗	✗	✗	✓
Interactive learning <sup>c</sup>	✗	✗	✓	✗
Our system CLEVER	✓	✓	✓	✓

**Table 6.1.:** CLEVER addresses several desiderata in developing a stream-based AL for real robots.

<sup>a</sup>i.e., Blum et al. (2022); Frey et al. (2022)

<sup>b</sup>i.e., Saran et al. (2023); Schmidt & Gunnemann (2023); Narr et al. (2016)

<sup>c</sup>i.e., Azagra et al. (2020); Valipour et al. (2017); Schiebener et al. (2011)

algorithm identifies the most useful unlabeled instances to learn from (Cacciarelli & Kulahci, 2024). In the literature, a pool of unlabeled instances is mostly assumed, resulting in the so-called pool-based AL (Noseworthy et al., 2021; Feng et al., 2022b; Akcin et al., 2023). In contrast, we focus on a setting in which data arrive in the stream, which is an underexplored area of AL (Cacciarelli & Kulahci, 2024). For robotic perception, the early attempts for stream-based AL relied on classical learning techniques such as Gaussian Processes (Triebel et al., 2016), boosting (Mund et al., 2015) and bagging (Narr et al., 2016). Yet, the current de-facto standard in object recognition relies on deep learning, urging for extensions of stream-based AL to DNNs. Although current extensions (Saran et al., 2023; Schmidt & Gunnemann, 2023) study the feasibility of stream-based AL using DNNs, the discussions therein are centered on strategies for informative data selection from the data stream.

Indeed, the central objective of AL is to reduce the cost of labeling by querying and selecting the most informative data (Cacciarelli & Kulahci, 2024). In contrast, our focus is applying AL to enhance the robustness of robotic perception by seeking human support and updating DNNs online. A similar use case was also previously mentioned by Triebel et al. (2016); Narr et al. (2016). However, due to a different focus, no real system was therein developed and evaluations were limited to showing sample efficiency, i.e., accuracy increase per newly added data points. Instead, we take a systems approach to the problem, thereby developing CLEVER that meets various requirements reported in table 6.1.

**Bayesian adaptation of neural networks.** Learning semantics from new streams of observation is a crucial capability for our system. In robotics, such adaptations with DNN have previously been investigated (Blum et al., 2022; Frey et al., 2022). Their findings suggest that continual learning during deployment improves the accuracy of the robot’s perception when compared to fixed, pretrained DNNs. However, their applicability to stream-based AL is limited, since no uncertainty estimates are available for the query and selection step of AL. In contrast, our work explores Bayesian methods that are well suited for stream-based AL within a unified framework. For this, we extend our previous work (Schnaus et al., 2023) to stream-based AL. We previously showed how continual learning can be performed while obtaining well-calibrated uncertainty estimates and generalization with DNNs using few data samples (Schnaus et al., 2023). We point out that such properties are desired for developing a complete stream-based AL system with real robots.

**Robot learning from humans.** In robotics, many works on learning from demonstration focus on mapping the states of robots to actions (Ravichandar et al., 2020). Among them, a recent work (Ren et al., 2023) shares a similar spirit to ours where a learning algorithm is designed to ask for human support using uncertainty estimates. However, their focus is on robot planning using large language models. An idea on the correction of DNNs with language instructions from humans is also being explored for policy learning (Shi et al.,

2024). Many researchers have investigated incremental and interactive learning of new objects using human instructions (Azagra et al., 2020; Valipour et al., 2017; Schiebener et al., 2011). Among them, we extend the works on a humanoid robot, ARMAR-III, where the robot demonstrated interactive learning of unknown objects from human instructions (Asfour et al., 2006; Schiebener et al., 2011). These early attempts showed that new rigid objects, such as books, can be learned using hand-crafted visual features and a k-nearest-neighbor classifier. Our extensions provide (a) the use of DNNs, (b) stream-based AL that reduces model uncertainty, and (c) fewer prior assumptions about the object.

### 6.3. System Concept and Challenges

Whenever uncertain, our system seeks human support and improves itself online based on human instructions. The main problem encountered is distribution shifts that the robot inevitably encounters during the deployment of a DNN-based model, which ultimately produces misleading and overconfident predictions. For example, a robot may face unknown objects. Imagine a robot trained to classify apples but spots bananas during deployment. Deformable objects present similar challenges if the induced deformation in the object’s shape is underrepresented in the training data. Given this problem, the concept of our system is to ask for help from humans and adapt the model online (see figure 6.1).

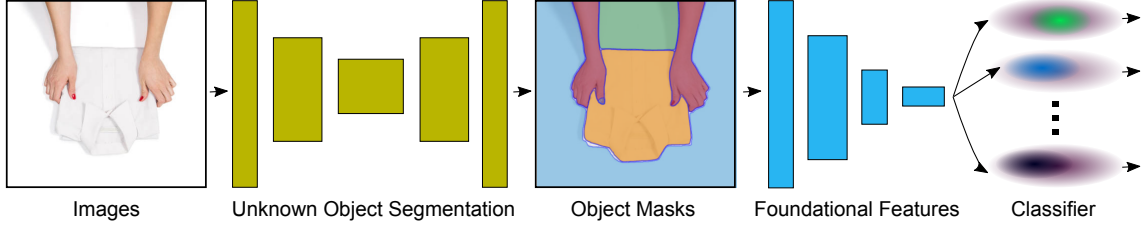
There are several challenges (see table 6.1). First, the system should know when the model is uncertain. This requires a probabilistic treatment that provides well-calibrated uncertainty estimates under distribution shifts. Given a training data  $\mathcal{D}$  and a DNN,  $f_{\theta}$ , where all learnable parameters are stacked as a vector  $\theta$ , probabilistic treatments of the given problem infer the posterior  $p(\theta|\mathcal{D})$  and compute a predictive distribution  $p(\mathbf{y}^*|\mathbf{x}^*, \mathcal{D})$  for new test datum  $(\mathbf{x}^*, \mathbf{y}^*) \notin \mathcal{D}$ . Second, the system should adequately support queries from the human and a decision-making framework that enables effective instruction of the robot by a human. Therein, acquired data cannot be of large amounts. Thus, we need an algorithm that generalizes robustly under small data conditions. Third, the model should continually learn efficiently and produce accurate predictions. For demonstrations on a humanoid, DNNs are to be trained fast, e.g., under one minute. To achieve this, CLEVER trains only a subset of model parameters  $\bar{\theta} \subseteq \theta$  so that the model does not forget the previous tasks. We also select a subset of data  $\bar{\mathcal{D}}^* \subseteq \mathcal{D}^*$  that is the most informative to learn. This reduces the training time as well as the model’s uncertainty. The latter minimizes the need for repeated human querying.

### 6.4. CLEVER: A Stream-based Active Learner

Having discussed the system concept and challenges, we describe the design of CLEVER.

#### 6.4.1. Underlying Prediction Model for Continual Adaptation

Our first component is a prediction model for semantic perception (see figure 6.2). The overall model takes images as input and outputs segmentation masks with their associated output labels. For this, we rely on an unknown object segmentation such as Segment-Anything (Kirillov et al., 2023) to generate the segmentation masks. A tracker can be combined to achieve a faster runtime of the model (Kirillov et al., 2023). Then, we stack the obtained masks and extract features using foundational models such as DinoV2 (Oquab et al., 2024). The obtained features are used as input of a classifier, which generates output



**Figure 6.2.:** Our prediction model for stream-based AL using DNNs. For semantic segmentation, our pipeline combines an unknown object segmentation (Segment-Anything), foundational representations (DinoV2) and a classifier based on Bayesian Neural Networks (BNNs) with multilayer perceptron. BNNs are represented as Gaussian distribution over the model parameters.

labels for each segmentation mask. In contrast to existing models such as Mask-RCNN, our construction allows open-set extraction of segmentation masks and state-of-the-art visual features using pre-trained models. This means we only need to train a classifier. In lieu of sophisticated pipelines for semantic segmentation such as Mask-RCNN, we focus on AL for classification tasks while also obtaining detection and segmentation results.

Next, we present the design of our classifier for stream-based AL. The classifier learns one multilayer perceptron (MLP) per object class. We call each of these MLPs the columns of our classifier. If we had an apple and a banana, our classifier would have two columns, each responsible for only one object class. In this way, a multi-class classification task is tackled using a combination of binary classifiers with calibrated uncertainties. The proposed construction brings two advantages. First, we can mitigate catastrophic forgetting by design. As we create a new column for a new incoming class, learning new objects does not affect the previously learned columns. Moreover, in each learning cycle, we can update only a column of our classifier. For example, if a column responsible for an apple exhibits high uncertainty, we can only learn to classify an apple better. Once the classifier is confronted with an unknown object, we can add and learn one new column. Training a smaller model can be more efficient. Achieving these results with a multi-class classifier might be difficult.

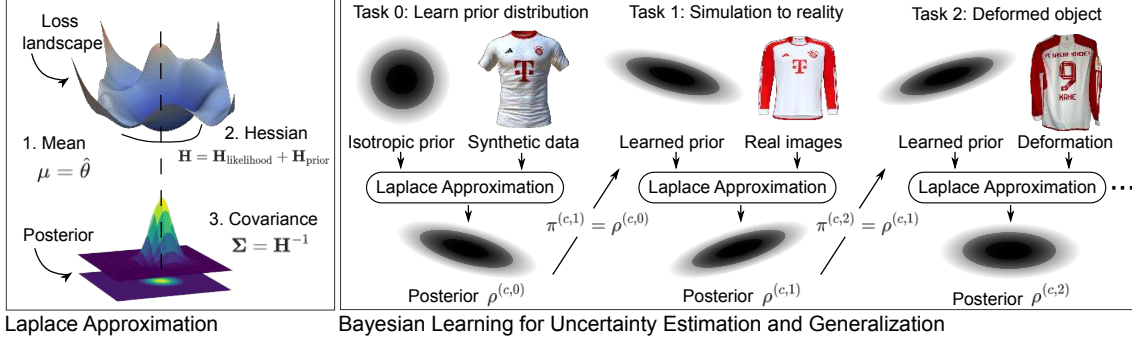
Within this construction, we apply probabilistic inference on all MLPs to obtain Bayesian Neural Networks (BNNs), which can provide well-calibrated uncertainty under distribution shifts. Let us define the training data for the classifier as  $\mathcal{D} = ((\mathbf{x}_i, \mathbf{y}_i))_{i=1}^N$ , where the inputs  $\mathbf{x}_i$  are the extracted features. We use superscripts  $(c, t)$  to denote the available  $C$  classes and  $T$  tasks, respectively. Then, a classifier  $f_{\theta}$  is now divided into several binary classifiers  $f_{\theta^{(c,t)}}$  that output  $\mathbf{y}_c \in \{0, 1\}$ . The corresponding DNN’s learnable parameters  $\theta^{(c,t)}$  belong to the column  $c$  for the task  $t$ , which is obtained using the relevant training data  $\mathcal{D}^{(c,t)}$ . BNNs apply probabilistic inference to DNNs, e.g., models are represented by the posteriors  $\rho^{(c,t)} = p(\theta^{(c,t)} | \mathcal{D}^{(c,t)})$ . BNNs predict through marginalization:

$$p(\mathbf{y}_c^* = 1 | \mathbf{x}^*, \mathcal{D}^{(c,t)}) = \int p(\mathbf{y}_c^* | \mathbf{x}^*, \theta^{(c,t)}) p(\theta^{(c,t)} | \mathcal{D}^{(c,t)}) d\theta^{(c,t)}. \quad (6.1)$$

Intuitively, instead of one model that best fits the data, BNNs pay tribute to the model uncertainty for predictions. The probabilities obtained better reflect the true belief in the class  $\mathbf{y}_c^*$  than the often-used softmax scores that tend to bias towards higher probabilities, or overconfident predictions (Lee et al., 2020b; Schnaus et al., 2023).

We then combine the calibrated binary classifiers using BNNs for handling the given multi-class classification task. Given a single test input  $\mathbf{x}^*$ , we obtain a vector of probabilities from all columns:  $\mathbf{p} = (p_1, p_2, \dots, p_C)$  where  $p_c = p(\mathbf{y}_c^* = 1 | \mathbf{x}^*, \mathcal{D}^{(c,t)})$ . Then, we pick the





**Figure 6.3.:** The proposed pipeline that learns a prediction model for stream-based AL. Left: Laplace Approximation (LA) infers the posterior distribution of a DNN as a Gaussian distribution. Right: Using LA to obtain BNNs and further exploiting training data that encodes our domain knowledge, we learn an informative prior from a posterior of a previous task. We note that our assumption is the existence of relevant task and data to learn the prior from. One example is utilization of synthetic data from photorealistic simulator. Another is to use generative models.

predicted class label that is most probable. We achieve this by:

$$\mathbf{y}^* = \operatorname{argmax}_c(\mathbf{p}) \text{ with } p(\mathbf{y}^* = c | \mathbf{x}^*) = p_{c=\mathbf{y}^*}. \quad (6.2)$$

In this way, we choose the corresponding class label and probability score. We do not consider one vs. rest multi-class strategy. Hence, the vector  $\mathbf{p}$  itself is not a valid probability distribution. For querying from humans and selecting the most informative data using uncertainty estimates, a valid probability distribution for the most likely class is sufficient. Thus, we do not require probabilities for all classes at once. Moreover, the given design choice enables our system to be more efficient when training because we do not need to update all the columns for training a single column – an assumption in one vs. rest multi-class strategy. Instead, we can update each column individually for efficiency.

#### 6.4.2. Bayesian Learning for Uncertainty and Generalization

We now present our training pipeline. Our goal is to obtain the posteriors  $\rho^{(c,t)}$  for uncertainty estimation. We also aim to address the challenge of generalization under a small data regime with DNNs. For this, our main idea is to learn the prior distribution over the parameters of DNNs. To explain, according to Bayes' rule, the posteriors are proportional to the likelihood  $\prod_{i=1}^N p(\mathbf{y}_i | \mathbf{x}_i, \boldsymbol{\theta}^{(c,t)})$  and the prior  $\pi^{(c,t)} = p(\boldsymbol{\theta}^{(c,t)})$ , i.e.,  $\rho^{(c,t)} = p(\boldsymbol{\theta}^{(c,t)} | \mathcal{D}^{(c,t)}) \propto \pi^{(c,t)} \prod_{i=1}^N p(\mathbf{y}_i | \mathbf{x}_i, \boldsymbol{\theta}^{(c,t)})$ . Due to the non-linearity of DNNs, no closed-form solution exists for the posteriors. Thus, we need approximate Bayesian inference – a set of algorithms that approximate the intractable posteriors. For this, the first step is to define the priors over the DNN parameters. Traditionally, a zero-mean isotropic Gaussian prior – an uninformative prior that regularizes the overall model – was seen to be sufficient for DNNs. However, when no large amounts of data are available, the likelihood no longer dominates the posterior. Thus, specifying an informative prior can improve the approximate inference (Schnaus et al., 2023). Figure 6.3 shows our pipeline, which we later combine with a generalization framework called the PAC-Bayes theory.

The first step of our pipeline is to learn an initial prior distribution offline using synthetic data (task 0 in figure 6.3). Synthetic data for object recognition can be generated by either photorealistic synthesizers such as BlenderProc2 or generative models such as StableDiffusion with relevant prompts like “A jersey on a table” or “Display Bayern Munich jersey”. Synthetic

data has the advantage that large amounts of annotated training data can be generated in a cost-effective manner. Hence, for all the known classes of objects, we generate training data  $\mathcal{D}^{(c,t=0)}$ . Now, in order to learn an initial prior, we apply Laplace Approximation (LA). LA is an approximation inference method that imposes Gaussian distribution on the DNN parameters around a local mode (Lee et al., 2020b). The obtained posterior has its mean as the maximum-a-posterior (MAP) estimates  $\boldsymbol{\mu}^{(c,t)} = \hat{\boldsymbol{\theta}}^{(c,t)}$ , which can be obtained using the standard DNN training procedure with a cross-entropy loss. In LA, the covariance of the posterior  $\boldsymbol{\Sigma}^{(c,t)}$  is estimated by an inverse of a loss landscape’s Hessian  $\mathbf{H}^{(c,t)}$ , i.e., a second order derivative of log posterior w.r.t the DNN parameters  $\boldsymbol{\theta}^{(c,t)}$ . By definition,  $\mathbf{H}^{(c,t)} = \mathbf{H}_{\text{likelihood}}^{(c,t)} + \mathbf{H}_{\text{prior}}^{(c,t)}$ . Assuming a zero-mean isotropic prior,

$$\text{Prior: } \pi^{(c,0)} = \mathcal{N}(\mathbf{0}, \gamma \mathbf{I}), \quad (6.3)$$

$$\text{Posterior: } \rho^{(c,0)} \approx \mathcal{N}(\hat{\boldsymbol{\theta}}^{(c,0)}, (\mathbf{H}_{\text{likelihood}}^{(c,0)} + \gamma \mathbf{I})^{-1}),$$

are the prior-and-posterior pairs. The posteriors at  $t = 0$  are then used as priors for task  $t = 1$ . We use a Kronecker factorization technique (Schnaus et al., 2023) to compute the Hessian in practice. More details about this technique can be found in chapter 3.

Having learned the priors, we now iterate the learning process online. Examples of incoming tasks are semantic segmentation on real images, or recognition of deformable objects, as depicted in figure 6.3. In each step, the approximated posteriors from the previous tasks are used as the priors for the current learning tasks. Intuitively, as we keep learning one object class per column, such Bayesian learning results in positive transfers across the tasks, i.e., posteriors that classify folded clothes help in learning unfolded clothes, even with small amounts of data. For this, we repeat LA and obtain:

$$\text{Prior: } \pi^{(c,t)} = \mathcal{N}(\hat{\boldsymbol{\theta}}^{(c,t-1)}, (\mathbf{H}^{(c,t-1)})^{-1}), \quad (6.4)$$

$$\text{Posterior: } \rho^{(c,t)} \approx \mathcal{N}(\hat{\boldsymbol{\theta}}^{(c,t)}, (\mathbf{H}_{\text{likelihood}}^{(c,t)} + \mathbf{H}^{(c,t-1)})^{-1}).$$

The prior-and-posterior pairs are obtained with small modifications to LA. First, the maximum-a-posterior estimates are computed using a more expressive prior, instead of initializing around zero with one variance for all DNN parameters. Second, for approximating the posteriors, the Hessian from the posterior of the previous task is used instead of an isotropic term  $\gamma \mathbf{I}$ . Our pipeline adapts our previous method Schnaus et al. (2023) to better fit our use case. Unlike previously, we do not attempt to transfer across columns. This design choice enables class-independent learning, which is more efficient. Moreover, we embed the domain knowledge with synthetic data at task 0, instead of foundational models.

For these steps of Bayesian learning, we can explicitly design for improving generalization of the model under small data regime. We achieve this by introducing hyperparameters  $\tau, \alpha$  and  $\beta$  such that  $\mathbf{H} = \tau(\beta \mathbf{H}_{\text{likelihood}} + \alpha \mathbf{H}_{\text{prior}})$ , and optimizing for a generalization objective called PAC-Bayes bounds. Note that we dropped the superscript  $(c, t)$  for notation simplicity when explaining PAC-Bayes theory. The hyperparameters  $\alpha$  and  $\beta$  decide how much weight should be given to the previous task (“prior”) and the current data at hand (“likelihood”). If more weight is given to the current data, the model may overfit, while more weight to the prior may result in underfitting. The tempering term  $\tau$  scales the overall posterior. We pick these hyperparameters by minimizing an upper bound to the expected loss  $\mathbb{E}_{\boldsymbol{\theta} \sim \rho}[\mathcal{L}_P^l(f_{\boldsymbol{\theta}})]$  on the true distribution  $P$ . A true expected loss incurs over the true data distribution  $P$  – not only on observed training and test data. Such generalization bounds depend on empirical loss on the training data  $\mathbb{E}_{\boldsymbol{\theta} \sim \rho}[\hat{\mathcal{L}}_{\mathcal{D}}^l(f_{\boldsymbol{\theta}})] = \frac{1}{N} \sum_{i=1}^N \hat{\mathcal{L}}_{\mathcal{D}}^l(f_{\boldsymbol{\theta}})$  and

```

begin
  // Initialization
   $\rho^{(c,0)} \leftarrow \text{prior}(\mathcal{D}^{(c,0)}, \pi^{(c,0)}) \forall c. ;$  // equation 6.3
   $\alpha^{(c,0)}, \beta^{(c,0)}, \tau^{(c,0)} \leftarrow \text{pac-bayes}(\cdot) \forall c. ;$  // equation 6.5

  // Main Loop
  while incoming image stream do
     $p_c | \mathbf{x}_k^* \leftarrow \text{marginalization}(\mathbf{x}_k^*) \forall c. ;$  // equation 6.1
     $p_c | \mathbf{x}_{1:k-1}^* \leftarrow \text{filter}(p_c | \mathbf{x}_k^*) \forall c. ;$  // equation 6.6 (optional)
     $\mathbf{y}^*, p_{c=\mathbf{y}^*} \leftarrow \text{prediction}(\mathbf{p} | \mathbf{x}_{1:k-1}^*) ;$  // equation 6.2
    if query humans for column  $c$  then
       $\mathcal{D}_{new}^{(c,t)} \leftarrow \text{human-instruction}(\cdot) ;$  // figure 6.1
       $\bar{\mathcal{D}}^{(c,t)} \leftarrow \text{selection}(\mathcal{D}_{new}^{(c,t)}) ;$  // equation 6.7
       $\rho^{(c,t)} \leftarrow \text{posterior}(\bar{\mathcal{D}}^{(c,t)}, \pi^{(c,t)}) ;$  // equation 6.4
       $\alpha^{(c,t)}, \beta^{(c,t)}, \tau^{(c,t)} \leftarrow \text{pac-bayes}(\cdot) ;$  // equation 6.5
    end
  end
end

```

**Algorithm 5:** CLEVER - stream-based active learner.

the KL-divergence of the priors and the posteriors. Our KL-divergence is dependent on  $\tau$ ,  $\alpha$  and  $\beta$ . For  $\epsilon > 0$ , the so-called PAC-Bayes bounds are defined as:

$$P_{\mathcal{D} \sim P^N}(\forall \rho \ll \pi : \mathbb{E}_{\theta \sim \rho}[\mathcal{L}_P^l(f_\theta)] \leq \delta(\mathbb{E}_{\theta \sim \rho}[\mathcal{L}_D^l(f_\theta)], \text{KL}(\rho \parallel \pi), N, \epsilon)) \geq 1 - \epsilon. \quad (6.5)$$

For more details, we again refer to chapter 3 where we devised a computationally tractable method of optimizing PAC-Bayes bounds for BNNs with LA.

### 6.4.3. A Temporal Active Learning System with Humans

The remaining challenges are to develop a system that (a) asks for help from humans and (b) selects the most informative samples to learn from. For both components, we combine the temporal information inherent in data streams.

Our query strategy involves a recursive rule that keeps the current probabilities about an object class given all measurements until step  $k$ :  $p(\mathbf{y}_c^* = 1 | \mathbf{x}_{1:k}^*)$ . Defining  $l(\cdot) = \log p(\cdot)[1 - p(\cdot)]^{-1}$  from which we can retrieve the current probabilities  $p(\cdot) = 1 - (1 + \exp[l(\cdot)])^{-1}$ , our recursive form based on a binary state Bayes filter is given by:

$$l(\mathbf{y}_c^* = 1 | \mathbf{x}_{1:k}^*) = l(\mathbf{y}_c^* = 1 | \mathbf{x}_k^*) + l(\mathbf{y}_c^* = 1 | \mathbf{x}_{1:k-1}^*) - l(\mathbf{y}_c^* = 1). \quad (6.6)$$

The obtained probabilities are converted to a normalized entropy as a measure of uncertainty, and we use a pre-defined threshold for query decisions (Triebel et al., 2016). Here, temporal information may filter the outliers and augment the tracking of object segmentation. Our design choice on utilizing binary classifiers per column also enables us to simplify the incorporation of temporal information. Instead of complex models such as Dirichlet, we only modified the algorithm behind the well-known probabilistic grid map (Burgard et al., 1996) that tracks uncertainty on binary states such as occupied or non-occupied space.



Next, given a human demonstration, AL selects the most informative data. For CLEVER, such a selection results in smaller training data, which makes the continual adaptation of the underlying model more efficient. We utilize the so-called BatchBald (Kirsch et al., 2019) to select the top  $\bar{B} \subset B^{(c,t)}$  data points:

$$\begin{aligned} \mathcal{A}_{\text{batchbald}}(\mathbf{x}_{1:B}^*, \rho^{(c,t)}) &= \mathbb{I}(\mathbf{y}_{c,1:B}^*, \boldsymbol{\theta}^{(c,t)} | \mathbf{x}_{1:B}^*, \mathcal{D}_{\text{new}}^{(c,t)}) \\ &= \mathbb{H}(\mathbf{y}_{c,1:B}^* | \mathbf{x}_{1:B}^*, \mathcal{D}_{\text{new}}^{(c,t)}) - \mathbb{E}_{\rho^{(c,t)}} \mathbb{H}(\mathbf{y}_{c,1:B}^* | \mathbf{x}_{1:B}^*, \boldsymbol{\theta}^{(c,t)}, \mathcal{D}_{\text{new}}^{(c,t)}). \end{aligned} \quad (6.7)$$

Intuitively, this acquisition function examines the mutual information  $\mathbb{I}$  between the multiple model predictions and the model parameters. Such mutual information is obtained by approximations to the entropy terms  $\mathbb{H}$ . The coupling between the model predictions for a batch of data points and the model parameters is captured, and data points with high mutual information would inform us more about the true model parameters. This allows us to again combine temporal information by considering a batch of data points. We further combine a sub-sampling strategy (Feng et al., 2022b), that is, we randomly select a subset from the demonstration data before applying BatchBald. This ensures the diversity of the samples while speeding up the computations of BatchBald.

#### 6.4.4. System Overview and Implementation Details

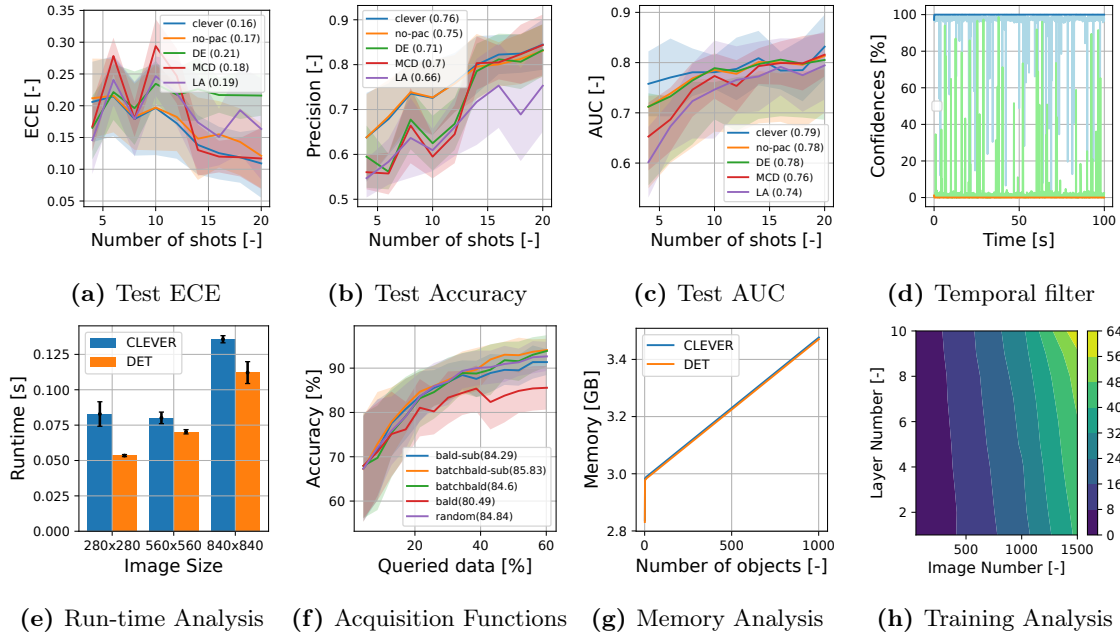
Algorithm 5 provides an overview of CLEVER, which shows how all the proposed components are integrated into a single algorithm. For each task, humans instruct the semantic information about an object using either speech or keyboard. Then, the human provides demonstrations about the given object from different viewpoints. Negative examples from the previous demonstrations are also included for training each column. For unknown objects, we add a new column and start the learning process with an isotropic prior. In algorithm 5,  $\bullet$  are used to indicate any arbitrary but relevant inputs to a function.

## 6.5. Experiments and Evaluations

### 6.5.1. Ablation Studies and Comparative Assessments

In this section, we provide ablation studies and comparative assessments to provide insights into the design of CLEVER. The analysis is focused on the continual learning architecture, the impact of using posteriors as informative priors, and combining temporal information. In particular, we focus on how our design choices address the challenges listed in table 6.1. For this, we collect a dataset from ARMAR-6, which consists of images from ten household objects. CAD models of these objects are obtained using an Android app called MagiScan. We also used StableDiffusion for synthetic data generation. For each object, we provided nine sequences of human demonstrations with 250 images each. We varied the difficulty level by providing more deformations to the objects, for example. We note that the dataset will be released to the public. There were no open-source datasets which fit our stream-based AL scenario due to the human element. We assumed that segmentation is given by foundational models like SAM, and mainly evaluate the underlying classifier.

**On continuously adaptable model.** First, we present our analysis on CLEVER’s ability to perform continual learning. In the beginning of the chapter, we claimed that the proposed design enables training of DNNs in less than one minute, while addressing the catastrophic forgetting problem. To evaluate the training time, we vary the number of layers from one to ten, and also vary the number of training data points up to 1500. Because we



**Figure 6.4.:** Results from ablation studies and comparative assessments of various design choices.

only train the MLP while fixing the representations from a foundational model, the results show that our classifiers can be updated in less than one minute (see figure 6.4h). We note that a three-layered MLP is used for all other experiments. Regarding the catastrophic forgetting, CLEVER adapts a progressive architecture where new columns are trained for new incoming object categories. By design, such architecture-based approaches mitigate the forgetting. However, growing the DNN architecture may hurt the computational scalability. To evaluate such computational scalability, we grow the DNN architecture to accommodate 1000 object categories. Comparisons in terms of memory and runtime are provided with CLEVER without probabilistic treatments (denoted DET). The results are depicted in figure 6.4g and 6.4e. Without an elaborated mechanism to mitigate catastrophic forgetting, CLEVER can learn 1000 object categories with less than 4GB of GPU peak memory and interactive frame rates. We also note that elaborated forgetting mechanisms can be introduced when an application scenario demands many more object categories.

**On Bayesian learning algorithm.** Secondly, we examine the influence of prior learning method for both uncertainty and generalization under small data regime. For this, we split the dataset into training and test set with a ratio of 8:2. Then, we randomly pick N-shot images per object category up to 20 images. Using Google uncertainty baseline (Nado et al., 2021), we evaluate various models (MC Dropout (Gal & Ghahramani, 2016) as MCD, Deep Ensemble as DE (Lakshminarayanan et al., 2017), Laplace Approximation as LA (Lee et al., 2020b), and no-pac means CLEVER without PAC-Bayes optimization (Schnaus et al., 2023)) with expected calibration error (ECE), accuracy and area under ROC curve (AUC) as the standard evaluation metrics. Chapter 2 discusses benchmarking practices. The findings are shown in figure 6.4c, 6.4b and 6.4c, where the chosen metrics are reported by averaging over the object categories. Five random seeds were used. The results show that CLEVER can outperform the chosen baselines. In particular, comparisons to LA and no-pac show that learning the prior from simulation, and optimizing for a generalization bound can improve the performance for the chosen evaluation scenario of stream-based AL.

	Success rate	ECE	Precision	Nr. Queries
CLEVERv1	<b>0.887±0.049</b>	<b>0.060±0.017</b>	<b>0.933±0.020</b>	<b>1.539±0.544</b>
CLEVERv2	0.826±0.053	0.092±0.036	0.900±0.034	2.440±0.866
Vanilla	0.801±0.054	0.177±0.021	0.817±0.039	4.320±0.992

**Table 6.2.:** Query success rate, ECE, precision, and number of queries to reach 85% accuracy are reported. Higher the better for query success rate and precision. Lower the better for ECE and number of queries. The baseline vanilla is the learner without the informative prior.

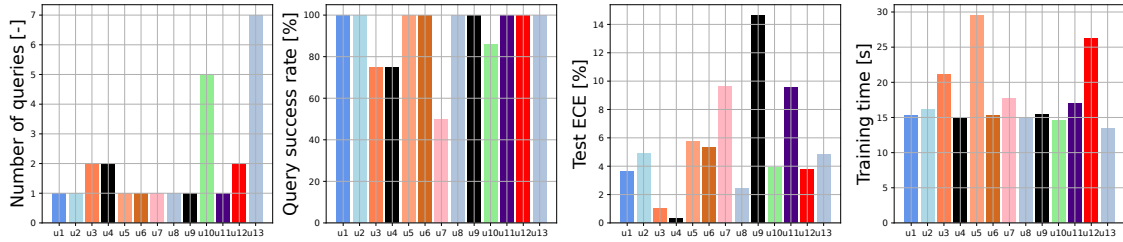
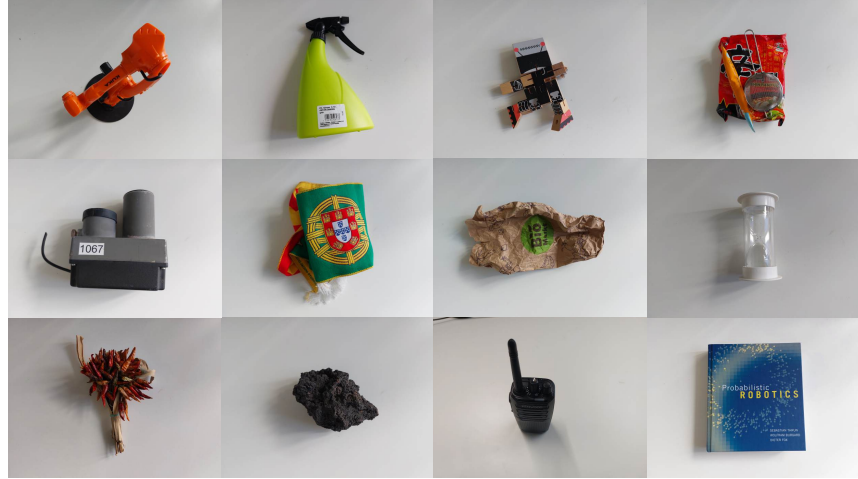
**On combining temporal information.** Third, we examine the idea of combining the temporal information when selecting the subset of images. This is to learn more efficiently by selecting the most informative samples, and also decide to query appropriately. For the former, we choose a uniform sampling, BALD, BatchBALD and their combinations with sub-sampling as baseline acquisition functions. For the latter, we train a model to provide noisy confidence estimates, and display filtered output against the raw output from a single query step of 100s. The same train-test split of 8:2 was used for the comparisons on acquisition functions. A total of 15 query steps were generated for all object categories. Five random seeds were used to obtain the results. For the combination of temporal data via filtering, we observe that noises can be removed (see figure 6.4d). Moreover, BatchBALD with the sub-sampling strategy, as the acquisition function looks at the batch of data points to measure information gain, outperformed the baselines in figure 6.4f. These findings motivate our design choice of combining temporal information for stream-based AL.

**Final performance evaluation.** Finally, we examine the final performance. Results are displayed in table 6.2. We assume that images arrive in a batch of streams over nine subsequent demonstrations with increasing levels of difficulty. Metrics of choice were query success rate, i.e., if the model queried correctly, average ECE and precision, and the number of queries required to reach more than 85 % precision. These metrics capture several requirements of a stream-based AL system. 40 data points were selected for training in each demonstration out of 250 data points so that the training terminates in less than a minute. Five random seeds were used. We compared three baselines. Vanilla corresponds to a deterministic model of CLEVER without any priors. Version 1 (CLEVERv1) used the full formulation of the prior with both mean and covariance, while version 2 (CLEVERv2) only utilized the mean by pretraining with the given synthetic data. We observe the gradual increase in performance in all metrics. These results justify our design choices, in particular, the prior learning with the targeted synthetic data for stream-based AL.

### 6.5.2. A Complete Open-set Evaluation with Users

Our goal is to evaluate the performance of CLEVER in an open-set condition, where the model encounters novel objects not from the training dataset. Furthermore, the goal is to perform user validation in order to show that many users can use the system successfully. To achieve this goal, we randomly invited thirteen users and asked them to bring any object of their choice. Figure 6.5 shows the example objects that were brought by the users to test the system. Since we did not know these objects a priori, we could create a truly open-set condition for CLEVER. During the user validation study, the users were instructed to use CLEVER in order to perform semantic perception of the objects they brought. Under these conditions, we measured the number of queries to learn the new object with more than 85% confidence, the number of query failures, test ECE, and training time. We let the user only collect 80 images per query step. Out of 80 images, CLEVER selected 32 images to

**Figure 6.5:** Examples of arbitrary objects brought by different users. The objects ranged from articulated, transparent, deformable, industrial and planetary objects. As these objects are unknown to CLEVER, semantic perception of these objects enables open-set evaluations.



**Figure 6.6.:** An open-set evaluation with 13 users (x-axis). Number of queries to semantically segment the objects with more than 85% confidence (lower the better), query success rate (higher the better), ECE as a measure of uncertainty (lower the better), and training time (lower the better) are reported from the experiments per user. These experiments validate that various users can perform active learning with CLEVER for semantic segmentation under open-set conditions.

adapt the model. In each query step, the posteriors of the previous task were used as the informative priors, along with an optimization of the generalization bounds.

The results are depicted in figure 6.6. The average number of required queries was  $2.0 \pm 1.79$ , while we had a query success rate of  $91.20 \pm 15.06\%$  where CLEVER associated well-calibrated confidence and appropriately asked for help. The average ECE was  $5.36 \pm 3.75\%$ , and CLEVER consumed  $17.79 \pm 4.717s$  training time. All users were able to work with CLEVER and perform stream-based AL under the replicated open-set condition. We also believe that a training time of less than 20s can be practical. Regarding limitations, small distributional shifts seem to be an issue. For example, we found it difficult to obtain well-calibrated uncertainty estimates when training a DNN with an apple but testing with an object similar to an apple in appearance, like a red pear. Nevertheless, all the objects brought by the user could be eventually conquered with CLEVER, which could have been difficult without adaptations at test time. In this sense, our experiments show the relevance of stream-based AL in developing a persistent vision system.

### 6.5.3. Demonstration on a Humanoid Robot

Finally, we demonstrate CLEVER on the KIT’s humanoid robot, ARMAR-6 [Asfour et al. \(2019\)](#). The scenario of demonstration is illustrated in figure 6.1. By doing so, we examine the feasibility of stream-based AL on a real robot. Three objects, namely an apple, a banana, and a T-shirt, are considered. ARMAR-6’s onboard cameras, speech interface, and NVIDIA

GeForce GTX 1080 are utilized (Asfour et al., 2019). Pre-designed language prompts were used to allow robot-human communication. The videos are on our project website. On ARMAR-6, we showcase CLEVER’s ability to perform robust semantic perception. We emphasize that, for all examined scenarios, deploying a standard DNN without any ability to adapt would have failed to complete the given perception tasks. In contrast, CLEVER is able to improve the robustness of deploying DNNs on a real robot. That is, CLEVER estimates uncertainties, asks for help from humans, and adapts itself to finally accomplish the given perception task. With these results, we empirically illustrate robust semantic perception by demonstrating the feasibility of stream-based AL on a real robot.<sup>2</sup>

## 6.6. Summary

In this work, we proposed CLEVER, a stream-based active learner for robust semantic perception with robots. Experimentally, ablation studies and comparative assessments are provided first in order to provide the insights behind the system. We also evaluated CLEVER in a complete open-set condition with a user validation study, in which participants brought various objects that are transparent, deformable, articulated, industrial, and planetary. We also showed how CLEVER can be integrated into a humanoid robot. In general, using CLEVER, these results suggest the possibilities of robust semantic perception, while embracing the predictive performance of deep learning. In future, improvements in unknown object segmentation techniques will also help our system. Thus, as a next step, we envision active learning on foundational models directly for open-set recognition tasks.

---

<sup>2</sup>All implementation details including metrics, synthetic data generation and in-depth discussions are in supplementary materials of our project website.



---

Learning Fluid Flow Visualizations from In-Flight Images with Tufts

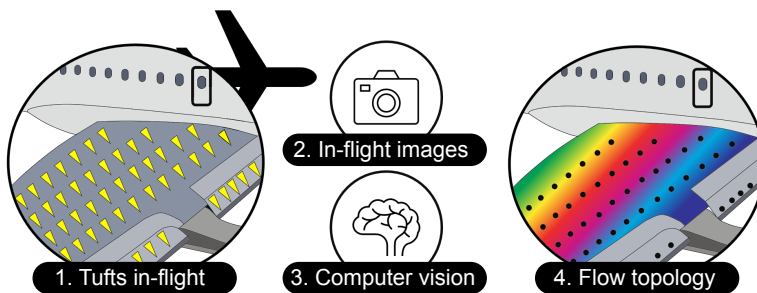
---

### 7.1. Motivation

Over the last decade, the performance of computer vision techniques improved significantly, leading to new application areas of robotics and automation. In particular, advances in deep learning introduced several frameworks capable of visualizing complex 3D flow phenomena (Liu et al., 2022). Such advances in fluid flow visualizations are relevant for aerial robotics, which requires a fundamental understanding of the underlying physics behind the flow phenomena. The efficient design of novel systems (Ruiz et al., 2022), safe operations of Unmanned Aerial Vehicles (Achermann et al., 2019; Muskardin et al., 2017), and the reduction of noise in rotary wing systems (Olsman, 2022) are a few examples that illustrate the relevance of the subject in current aerial systems research.

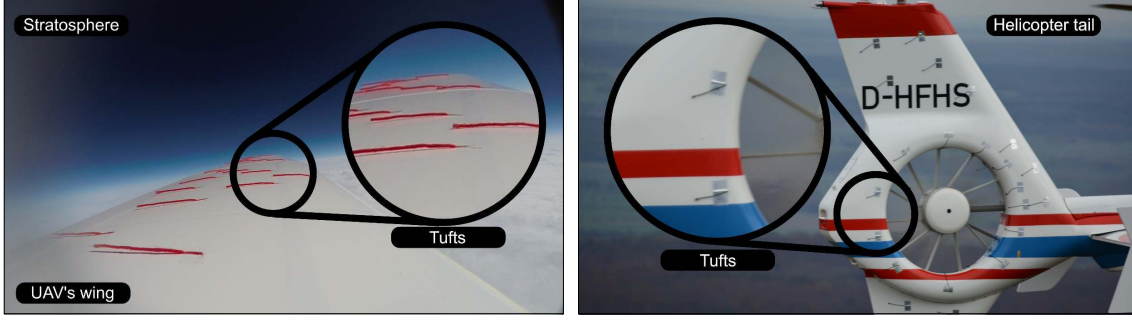
In this chapter, the goal is to advance our understanding of fluid flows around complex aerial systems in real flights. To do so, we focus on visualizing their local fluid flow topology via the application of tufts, one of the oldest and simplest experimental methods to visualize the flow on a surface. Tufts and their variants, such as flow cones, are small pieces of wire or rope that are attached to a surface, aligning themselves with the local flow velocity (Fisher & Meyer Jr, 1988). Typically, the orientation and shape of the tufts are captured with a camera and the images are analyzed to obtain insight into the local flow topology on the surface (Wieser et al., 2016; Vey et al., 2014). Examples are notably found in fluid dynamics literature (Wieser et al., 2016; Steinfurth et al., 2020; Vey et al., 2014).

In particular, we propose a learning system to advance the applicability of tufts for *in-*



**Figure 7.1:** Tufts are placed on an in-flight vehicle such as aircraft. Visual sensors capture the video sequences, and semantic segmentation of tufts is performed with learning algorithms for visualizing the local flow topology.





**Figure 7.2.:** Tufts are visualized and highlighted for the considered application scenarios. Left: helicopter tail rotor (the Fenestron duct). Right: wing area of a UAV, captured approximately 20 km altitude. Some tufts are zoomed in. Different class labels are given to each individual tuft in order to capture unsteady aerodynamics. The goal is semantic segmentation of these small and repetitive objects with similar appearance but different class labels. With semantic segmentation, this work enables the automation of evaluating images with tufts for experimental aerodynamics.

*flight fluid flow visualizations.* Given in-flight images with tufts, our system automatically generates semantic segmentation masks of individual tufts for flow visualizations (see figure 7.1). In contrast to a manual analysis by expert humans (Fisher & Meyer Jr, 1988), the proposed system enables automation. Such automation has the added benefit of flexibility, reproducibility, lack of human bias, and scalability to the vast number of images and tufts per image. Moreover, this concept is not restricted to controlled environments, where existing techniques of model-based computer vision such as template matching, homography, and the application of static masks may be sufficient (Vey et al., 2014; Steinfurth et al., 2020). The proposed system thereby scales the applications of tuft methods to challenging environments outside laboratories, where large changes in lighting conditions, perspective, and background scenes are inevitable.

The development of such a system is motivated by a large amount of challenging data we collected for concrete applications of flow visualization. Our flight data consists of (a) in-flight images from the tail surface of a helicopter, which we collected using another manned helicopter flying in close formation, and (b) the images from a UAV flying in the stratosphere, i.e., the flight sequence back and forth from ground to approximately 20 km altitude. Importantly, these images are challenging for model-based computer vision techniques due to the large variations between the different images. Moreover, annotated training data is not readily available for semantic segmentation. The problem also involves classifying small objects, that is, tufts, which have the same appearance but different labels (see figure 7.2). Therefore, in this use case, the applicability of end-to-end supervised deep learning techniques, such as Mask-RCNN, is also severely limited.

To this end, we propose a semantic segmentation pipeline that addresses the aforementioned challenges. The key idea is to divide the problem of semantic segmentation into three steps, namely (a) active learning-based object detection using uncertainty sampling, (b) uncertainty-driven image matching for object classification, and (c) weakly supervised instance segmentation with so-called uncertainty maps. In the chapter, we describe in detail how these probabilistic approaches allow the training of the overall pipeline without requiring any annotations of semantic segmentation masks. Instead, we reduce the annotation efforts to the labeling of fewer bounding boxes and only a single image where all the class labels are specified. Empirically, several comparative assessments are presented within these three steps, which motivate our design choices. Lastly, quantitative and qualitative



assessments of the proposed concept are provided as final demonstrations. We note that our focus is on examining the feasibility of computer vision techniques for the proposed application concept, while aerodynamic results will be presented in a future publication.

**Contributions and major claims.** In summary, the main contributions of this work are (a) a novel application concept of learning flow visualizations from in-flight images with tufts, (b) a semantic segmentation pipeline, based on probabilistic approaches, to address the practical challenges of our scenario, (c) the collection and sharing of in-flight images from the manned helicopter and stratospheric UAV flights, and (d) several experimental results to validate each component of the system, including quantitative and qualitative characterizations of the overall performance. We claim that our probabilistic approach enables semantic segmentation with fewer bounding box annotations and only one image where all the class labels are labeled. To the best of our knowledge, we are the first to provide real-world demonstrations of the proposed concept for fluid flow visualization.<sup>1</sup>

## 7.2. Related Work

We start by introducing traditional flow visualization techniques in aerodynamics. Then, we review existing works that apply image processing for automatic tuft recognition. As our probabilistic approaches reduce annotation efforts, we further discuss related work therein.

**In-flight flow visualization.** Due to the transparency of fluids, their flow patterns are invisible to humans. Thus, several experimentation methods have been devised to visually acquire the flow patterns. In wind tunnel facilities, which replicate the interaction between air and flight models by blowing air using large tubes, certain mediums such as smoke, oil flows, particles, etc. have been combined with optical measurement techniques to visualize the air flows. Likewise, such technologies are being applied to aerial systems in-flight as a way to obtain data from real flights. Here, flow cones and tufts, oil flows, liquid crystals, sublimating chemicals, and smoke are often used (Fisher & Meyer Jr, 1988). Among these methods, for simplicity and low cost, this work automatizes tuft methods.

**Automatic tuft recognition.** Over the past decades, the applications of tufts were mostly on qualitative flow visualizations in which images are manually analyzed by aerodynamicists (Fisher & Meyer Jr, 1988). Recently, however, quantitative analysis using image processing has also been examined. Vey et al. (2014) and Wieser et al. (2016) manually specified the position of each tuft and its geometric orientation and used mean angles to perform line integration convolution. A rule-based system is developed, where the masks and anchors of the surface that contain tufts, threshold-based foreground extraction, and color-based identification schemes are used (Chen et al., 2020). Steinfurth et al. (2020) assumed the location of each tuft to be known and applied the Prewitt method for shape extraction. These works show that the acquisition of quantitative information from tufts is a viable option. Our main novelty is a learning-based solution, which relaxes the assumptions of rule-based systems. With this, we demonstrate the real-world applicability at the scale of real manned helicopter flights and stratospheric missions of a UAV.

**Reducing the annotation efforts.** End-to-end semantic segmentation models heavily rely on large amounts of annotated training data, which are often not readily available. In such cases, the generation of synthetic data is a compelling option, where annotations are readily available. However, tufts are deformable, and hence the supports for such objects are limited along with the well-known problem of the sim2real gap (Feng et al., 2022b).

<sup>1</sup>Link to the project website: <https://sites.google.com/view/tuftrecognition/>.

Other alternatives are semi-supervised models (Mittal et al., 2019), which utilize only a small number of annotations, while weakly supervised models (Zhang et al., 2020a) use some weaker form of labels. In our approach, we are heavily based on advances in these two domains. The proposed three-step pipeline exploits active learning for bounding-box detection as a foreground extractor of tufts. Here, a small number of single-class bounding boxes are relatively easier to annotate. Then, image matching is devised by combining classical image matching, uncertainty estimates (Lee et al., 2020b), and the idea of keyframes for multiclass semantics. The final step involves instance segmentation, where we employ weakly supervised learning models (Zhou et al., 2018). In this way, we demonstrate how probabilistic approaches help in real-world applications with limited annotated data.

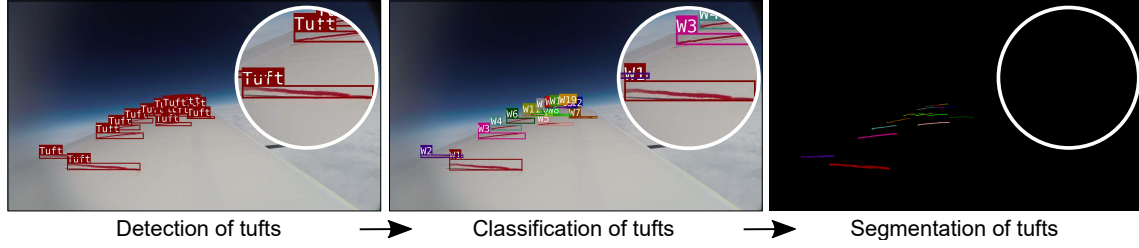
### 7.3. Data, Task, and Challenges

This section provides brief information on data collection, the task, and associated challenges.

**Data generation.** We performed flight experiments with the EC135-ACT/FHS helicopter equipped with 56-81 tufts, while the Bo105 helicopter was used as a camera platform. The resolution of captured images was 6000x4000 pixels. The goal of this first scenario is to analyze the complex aerodynamic behavior in flight of the anti-torque device known as Fenestron or fan-in-fin (Olsman, 2022). As a second scenario, we gathered data from a stratospheric UAV (Wlach et al., 2015) with 19 tufts on its wing. These images were captured with a GoPro during the approximately 145-minute long flight to the stratosphere, i.e., the atmosphere at about 20 km altitude. This dataset is to show the generality of our method. Stratospheric flight is also an area where the flow phenomena cannot be completely reproduced in wind tunnels (Lee et al., 2018b). We refer to the project website. More details on flight experiments including data collection setup have been included.

**Task definition.** From flight testing, we typically acquire thousands of images under different conditions, from which useful aerodynamic data can be extracted. Such information can be obtained automatically by a processing methodology that recognizes the exact shape of the tuft in pixels. Moreover, to examine the temporal behavior of each individual tuft, the same tuft in many images is to be classified. In other words, we are interested in the flow topology of an aerial system, and so each individual tuft must be monitored separately over time. Therefore, from a computer vision perspective, the problem involves semantic segmentation as described in figure 7.2. We note that after the images have been processed, aerodynamic results can be presented as streamlines on the surface (Wieser et al., 2016) or polar histograms for individual tufts (Vey et al., 2014). Here, streamlines are the velocity vector fields of airflow, while the polar histograms of the airflow directions are for quantitative data. Therefore, fulfilling the herein-defined computer vision task allows fluid flow visualization as well as quantitative characterizations.

**Challenges with flight data.** Observing the collected data from real flights, we find that rule-based systems (Vey et al., 2014; Wieser et al., 2016; Chen et al., 2020; Steinfurth et al., 2020) cannot be first applied directly, because we cannot assume known masks and anchors due to large variations in perspectives, relative positions, and illuminations between images. Clearly, if the sizes and locations of foreground masks change between the images, we cannot assume static masks and anchors. These large variations in perspective and position are caused by different relative positions of the following helicopter, to allow a view inside the Fenestron duct. Positional shifts also occur due to unintended movements of the following helicopter by turbulence and gusts. Although supervised learning techniques are the current gold standards for semantic segmentation, generating large amounts of



**Figure 7.3.:** An overview of our pipeline. Instead of supervised learning, using deep models, we devise a semantic segmentation pipeline with (a) detection of tufts for foreground extraction, (b) classification for semantic information and then (c) instance segmentation. We note that deep learning-based object detection and segmentation methods can also classify different objects. However, in our real-world application scenario, training data are not readily available, and labeling annotation masks is very costly. Thus, the proposed pipeline is to reduce the annotation efforts down to labeling of fewer bounding boxes, and only a single image where all the class labels of each tuft are specified, i.e., no annotations of semantic segmentation masks are required.

manual annotations is not feasible; for example, one could imagine annotating semantic segmentation masks for thousands of images as in figure 7.2. Hence, the direct applicability of end-to-end supervised deep learning techniques is limited. Lastly, the problem involves classifications of objects (tufts) with similar appearances, but different labels depending on their relative locations. To address this, we conceive that combining model-based techniques may be a solution rather than learning such patterns from data only.

On the other hand, we assume that the data are acquired either from videos or similar image capturing. Furthermore, the analysis is intended for offline and therefore, the run-time or efficiency is not an important criterion as far as our pipeline is computationally tractable. In contrast, system-level requirements are (a) to maximize accuracy and (b) to reduce manual efforts in producing annotations. Given these primitives, the next section presents our solution to the aforementioned practical challenges from real flights.

## 7.4. Probabilistic Tuft Segmentation Pipeline

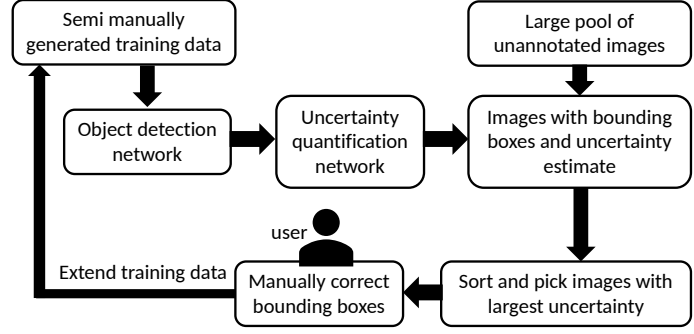
As depicted in figure 7.3, we propose to break this challenging problem into three feasible sub-problems. Next, we present our solutions to each individual problem.

### 7.4.1. Tuft Detection

In this step, the objective is to locate the presence of objects with a bounding box and further classify the located object in an image. This task is typically performed by a neural network trained in a supervised manner. Hence, bounding-box annotations are required for training. Although generating such annotations is more affordable than labeling for semantic segmentation, our goal is to reduce the manual human efforts as much as possible.

To this end, we propose a pool-based active learning framework with coarsely annotated pool data. An overview is provided in figure 7.4. In the first stage, we manually annotate a single image and propagate this annotation to other images via image matching. Like this, we automatically generate a large pool of coarsely annotated images. For some images, this will work; for many, it will not. In any case, it only requires manual correction by a user instead of generating annotations from scratch. With these initial training data for supervised learning, an object detector can be trained. Here, we simplify the detection

**Figure 7.4:** The proposed pool-based active learning with coarsely annotated pool data. The initial coarse annotations are generated manually with the aid of feature-based image matching.



task by leaving out the classification task, i.e. we treat different tufts as a single class “Tuft”. This avoids the problem of detecting objects with the same appearance but different class labels and increases the number of instances of supervised data points. Moreover, the annotation of the bounding boxes for a single class is less time-consuming.

The next stage then involves an active learning loop. Here, our algorithm queries a user for new annotations from a pool of data. Often, the subset of data is queried/selected with measures of uncertainty (Gawlikowski et al., 2023), i.e. the most uncertain data to neural networks are prioritized (Feng et al., 2022b). With the newly obtained data, the existing training data is extended and the network is re-trained. Repeating the loop, active learning automatically selects the most informative data for the network to learn from. The proposed pipeline uses coarsely annotated pool data as shown in figure 7.4. As such, coarse annotations are error-prone; our pipeline involves a user for corrections only when selected by the active learning algorithm. In this way, we reduce human supervision more. What further motivates active learning is the proposed simplification of object detection into a single class. As such, active learning is well-suited in this formulation of binary classification since we can avoid under-performance in multi-label setups (Feng et al., 2022b).

Concretely, two crucial components are uncertainty estimation for neural networks and a selection criterion for label query. The former is a framework, which is based on Bayesian formulation (Gal & Ghahramani, 2016; Lee et al., 2020b). Given any trainable parameters of neural networks  $\theta$  and training data  $\mathcal{D}$ , these frameworks estimate the weight posteriors  $p(\theta|\mathcal{D})$ . Then, the prediction uncertainty  $p(\mathbf{y}^*|\mathbf{x}^*, \mathcal{D})$  can be quantified:

$$p(\mathbf{y}^*|\mathbf{x}^*, \mathcal{D}) = \int p(\mathbf{y}^*|\mathbf{x}^*, \theta) p(\theta|\mathcal{D}) d\theta, \quad (7.1)$$

for a output  $\mathbf{y}^*$  from a new input  $\mathbf{x}^*$ . If applied to object detectors, we obtain a probabilistic object detector (POD) (Harakeh et al., 2020) that delivers the calibrated probabilities of class labels and the covariance matrices of the associated bounding box locations. This information is then used in generating the label queries. For this, we utilize the information scores for both classification  $\mathcal{U}_{j,cls}$  and regression  $\mathcal{U}_{j,reg}$  tasks respectively for all object instances  $j$  in an image (Feng et al., 2022b):

$$\mathcal{U}_{j,cls} = \sum_{i=1}^{|C|} \mathcal{H}(p(c_i|\mathbf{x}^*, \mathcal{D})) \quad \& \quad \mathcal{U}_{j,reg} = \mathcal{H}(p(\mathbf{b}|\mathbf{x}^*, \mathcal{D})),$$

which rely on the Shannon Entropy  $\mathcal{H}(\cdot)$  by assuming categorical distributions over the classes  $c_j$  and Gaussian distributions for the bounding boxes  $\mathbf{b}$  (a set of two-pixel coordinates describing a box). These scores can then be aggregated per image to select the most uncertain data. Aggregating is needed because we want to select the most informative images.

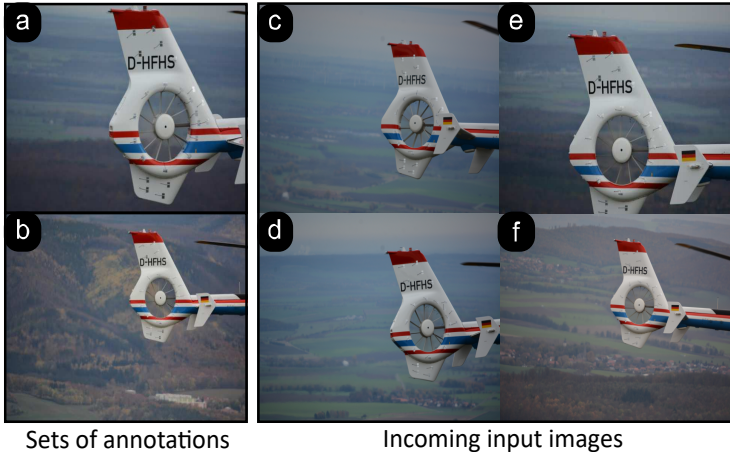
```

input :  $\mathbf{I}_T$ : the current test image from a video stream;  $p(\mathbf{y}^* | \mathbf{x}^*, \mathcal{D})$ ,  $\mathbf{b}$ : outputs of
        probabilistic object detection;  $\mathbf{x}^* = \mathbf{I}_T$  and  $\mathbf{b} \leftarrow \mathbb{E}[p(\mathbf{y}^* | \mathbf{x}^*, \mathcal{D})]$   $\{\mathbf{I}_{S,i}\}_{i=1}^K$ : The
        keyframes as a set of source images with annotations;  $L$ : The total number of tufts.
output :  $\{c_j\}_{j=1}^L$ : The class labels for each bounding box;  $\{\mathbf{I}_{T,i}\}_{i=1}^{K+1}$ : New key-frames after
        evaluating results on the current image  $\mathbf{I}_T$ .

begin
    /* Key-frame based Image matching */
    for all the K images in the key-frames do
        |  $\mathbf{T}_i, C_i \leftarrow \text{image\_matching}(\mathbf{I}_{S,i}, \mathbf{I}_T) \forall i.$  ; // Image matching
    end
     $\mathbf{T} \leftarrow \arg \min(\{C_i\}_{i=1}^K).$  ; // Select the result with the least cost
     $\{c_j\}_{j=1}^L \leftarrow \text{image\_matching}(\mathbf{b}, \mathbf{T}).$  ; // Label all L bounding boxes using Hungarian
    algorithm
    /* Is the results reliable? Multi-criteria decisions (MCD) are based on (a) If
       all L tufts are detected, (b) If the confidence is high, and (c) If the
       matching costs are low. */
     $\text{RS} \leftarrow \text{MCD}(p(\mathbf{y}^* | \mathbf{x}^*, \mathcal{D}), C_i, L).$  ; // Evaluate the Reliability Score (RS)
    if RS is True then
        |  $\{\mathbf{I}_{T,i}\}_{i=1}^{K+1} \leftarrow \text{update\_keyframe}(\{\mathbf{I}_{S,i}\}_{i=1}^K, \mathbf{I}_T, \{c_j\}_{j=1}^L, \mathbf{b}).$  ; // Update if reliable
        | more than a threshold
    end

```

Algorithm 6: Tuft classification algorithm with image matching



**Figure 7.5:** Given an annotated image as a source image, we propagate the labels to the target image via image matching. Due to various perspective changes, having multiple source images with annotations can increase the accuracy of image matching. For example, image a would match image e easier than image b. Likewise, image b can easily match image f.

### 7.4.2. Tuft Classification

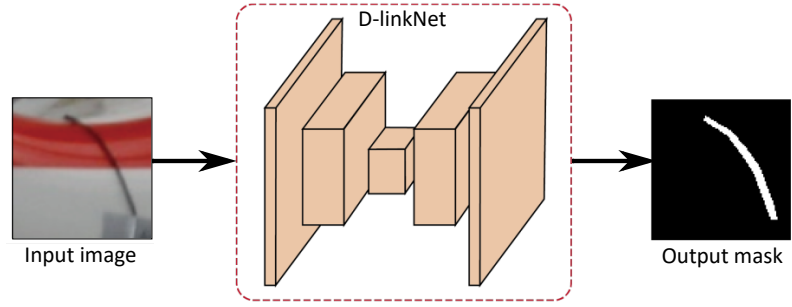
Having obtained the bounding boxes of all tufts with a single class “Tuft”, the goal is to classify them into their unique identification labels, like “Tuft W1”, “Tuft W2”, etc.

To achieve this, we devise an uncertainty-driven image matching. We provide a complete description of the pipeline in algorithm 6. Intuitively, given an annotated image that contains all the class labels, we can propagate the labels by means of matching this annotated image to the current test image. Due to the temporal nature of our data (such as videos), we avoid learning the classification of objects with a similar appearance. Yet, we have to deal with certain challenges such as image matching that can fail under severe perspective changes, illumination, and so on (see figure 7.5). Our algorithm is designed to address such challenges while still reducing the annotation efforts down to one single image.

Specifically, in algorithm 6, the inputs are the current test image  $\mathbf{I}_T$  and their corresponding outputs of probabilistic object detection  $p(\mathbf{y}^* | \mathbf{x}^*, \mathcal{D})$ ,  $\mathbf{b}$ , and the outputs are class



**Figure 7.6:** Tuft segmentation. We predict the segmentation masks using D-LinkNet. The network is trained in a weakly supervised manner.



labels for each bounding box  $\{c_j\}_{j=1}^L$ . Additionally, we use the so-called key-frames, i.e., a set of available images with annotations  $\{\mathbf{I}_{S,i}\}_{i=1}^K$ . For this, we initially annotated a single image:  $K = 1$ , and updated it until a specific desired number of annotated images is reached. As to reduce the manual annotations by humans, the key-frame update will be performed automatically. That algorithm achieves this using a criterion that assesses the reliability of automatically generated annotations. We stress that annotations with multiclass labels are more expensive than single-class labels with bounding boxes only. Finally, we take the total  $L$  number of tufts as another input since, in our application scenario, the number of tufts on an aerodynamic vehicle is known.

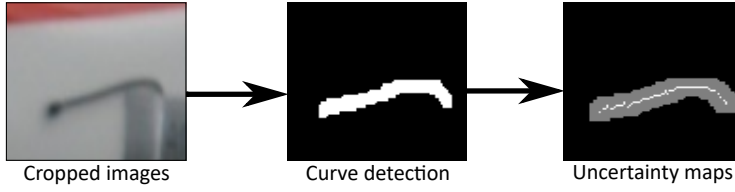
For the main body of algorithm 6, we first match the current image with images from the keyframes, which contain annotations of class labels. This results in multiple transformations and their costs, one pair each per source image. We pick a single source image  $\mathbf{I}_{S,i}$  with the lowest matching cost or error. Then, using this, we propagate the labels by keeping the available boxes from the input, while obtaining the multi-class labels by examining the overlapping areas and nearest points from the matching results. We use a Hungarian algorithm to assign each tuft one and exactly one label. This results in the first output of the algorithm:  $\{c_j\}_{j=1}^L$ . As the final step, we project the obtained results using three criteria, namely the number of detections, uncertainty of the bounding box predictions, and the error of image matching. These three criteria capture the reliability of the obtained results, and the keyframes are updated only when the results are deemed more reliable than a pre-specified threshold. Intuitively, the obtained results can be re-used as the source image for matching with the next images, only if deemed reliable from quantified uncertainty.

In this way, the idea of keyframes mitigates the failures of image matching, as the keyframes consist of multiple source images from different conditions. By further combining a decision-making criterion, we can automatically generate source images within the keyframes. This reduces any manual annotations of costly multi-class labels.

### 7.4.3. Tuft Segmentation

Given the locations of each tuft and their class labels, we now perform instance segmentation. So, within the cropped images from each bounding box, we group the pixels that belong to the object, separating the objects of interest from the background. This differs from applying the segmentation method on the whole image directly.

Our pipeline involves a network that predicts a segmentation mask of tufts given a cropped input image (see figure 7.6). To train this network, we first employ a line extraction via a curve line detector (Steger, 1998), which generates a coarse result. Then, D-LinkNet (Zhou et al., 2018) is learned from the coarse segmentation as a weaker form of annotation. In this way, we can perform instance segmentation without annotations of segmentation masks.



**Figure 7.7:** Generation of annotations for tuft segmentation. A classical curve detection is combined with uncertainty masks (gray). The used loss function does not use any pixels within the uncertainty maps.

First, a curve detection (Steger, 1998) is adopted because it extracts curvilinear structures by utilizing an explicit model for lines and their surroundings, as opposed to a simple model of only a line. Then, we train D-LinkNet, which has been originally developed to extract long and thin objects in the context of remote sensing. The architecture adopts the widely used encoder-decoder structure with dilated convolutions to connect the pre-trained encoder to the decoder, while the intermediate layers use a series of dilated convolutions in order to deal with tiny objects in an image and maintain the spatial details at each scale. The network is trained using the scribble-based weakly supervised loss function (Wei & Ji, 2021), with the partial binary cross-entropy (PBCE) loss and the auxiliary loss (AUXL):

$$L(\mathbf{Y}, \mathbf{S}) = \text{PBCE}(\mathbf{Y}, \mathbf{S}) + \text{AUXL}(\mathbf{Y}, \mathbf{S}), \quad (7.2)$$

where  $\mathbf{Y}$  is the curve mask from the curve detector and  $\mathbf{S}$  is the prediction mask of the deep learning model. In particular, the PBCE loss is defined as,

$$\text{PBCE}(\mathbf{Y}, \mathbf{S}) = - \sum_{x \in \omega} \mathbf{Y}_x \log(\mathbf{S}_x) + (1 - \mathbf{Y}_x) \log(1 - \mathbf{S}_x),$$

where  $x$  stands for a pixel in  $\omega$  - the regimes that are known, i.e. the pixels that belong to either the background or the object. This is provided by the curve detector, but we also apply a buffer along the detected line and set it as an unknown regime (see figure 7.7). As the PBCE loss is defined in the regimes that are known as foreground and background, the network learns to generate these regimes by minimizing the loss. Intuitively, the network can learn from examples where the curve detection performs well in order to separate between the foreground and background of tufts, and such patterns of lines persist within the unknown regimes. Finally, an additional loss function, AL, is used for improvements, which we ablate within our experiments in section 7.5.

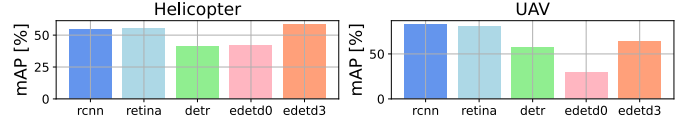
## 7.5. Experiments and Evaluations

Now, our quantitative results are presented for each individual step. The final performance is then examined. The implementation details can be found on our project website.

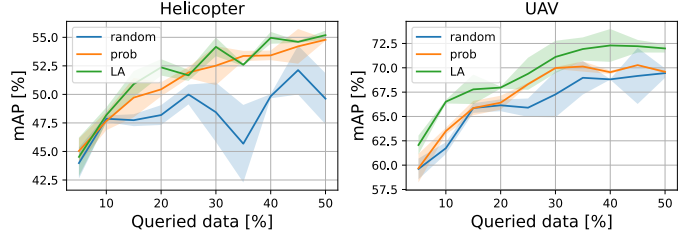
### 7.5.1. Results on Tuft Detection

We examine whether certain object detectors are more suitable for the given data. For this, we manually annotated 1000 images each for both the helicopter and the UAV data, and randomly chose 500 data points for training and the other 500 images for testing. For evaluation, the commonly used mAP is used. The selected models are RCNN, RetinaNet, efficientDet-v0 (edetd0), efficientDet-v3 (edetd3) (Tan et al., 2020) and DETR (Carion et al., 2020a), which are all widely applied detectors. We use open source code from Pytorch and Detectron2 and the models are trained with the ResNet backbone. We used a batch

**Figure 7.8:** The results of object detection for five different methods. Higher the better for mAP metric.



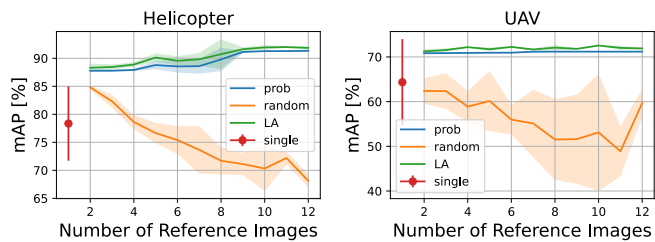
**Figure 7.9:** The results of active learning for different sampling strategies. The higher the better for the curves.



size of two and a learning rate of 0.001 and 0.01 was chosen for the helicopter and the UAV data, respectively. The results of model comparisons are shown in figure 7.8. We find that edetd3 produces the highest mAP for helicopter data, while RCNN achieved the highest mAP for the UAV data. While the results of each model differ depending on the data, we find that DETR and edetd0 underperform as they are not designed for small objects. In our scenario, we find that detectors for small objects with adaptable anchors are desirable.

For active learning, we use the same setup as the model comparison experiments, e.g., the metric, annotations, etc. However, the 0.2, 0.7 and 0.1 ratio splits are chosen for the test set, the pool set, and the validation set, respectively. A total of ten active learning loops are used to query up to 50 percent of the total pool data. Three random seeds are repeated for different initializations of neural network training. For the baselines, we used three different strategies. These are randomized selection from the pool set (random) and uncertainty-based sampling with the state-of-the-art uncertainty quantification methods, namely Monte Carlo dropout (prob (Gal & Ghahramani, 2016)) and Laplace Approximation (LA (Lee et al., 2020b)). These baselines are used to examine the influence of uncertainty estimates. Moreover, since we simplified the task to detect only one single class, uncertainty estimates are one of the influential variables to examine. Active learning results are reported in figure 7.9. Here, the queried data are represented as percentages of the total number of data. The error bars are depicted in shades. In general, we examine that random selection is outperformed by other methods, while LA results in superior performance. This motivates the design choice of our method. In addition, the results show that about 50% of the overall data would result in 95% of total performance. This is due to redundancy in the data and motivates the use of active learning to reduce annotations.

**Figure 7.10:** The results of tuft classification using the proposed image matching. The higher the better the curves.





### 7.5.2. Results on Tuft Classification

For evaluating the tuft classification task using uncertainty-driven image matching, we use the previously annotated 1000 images each for both the helicopter and the UAV dataset. Here, only one multiclass annotated image is used as the source, and other keyframes are chosen with our decision-making criteria<sup>2</sup>. Then, all the other images and annotations are used for evaluation. The mAP metric is chosen, which captures both the classification accuracy and the refinement of the bounding box. The number of keyframe images is ablated from two to twelve in order to show that additions of keyframes enable more accurate image matching. To evaluate the selection of keyframes, we compare our probabilistic approaches to a random selection, which forms a baseline to show that careful selection of keyframes can increase the accuracy of the proposed image matching algorithm.

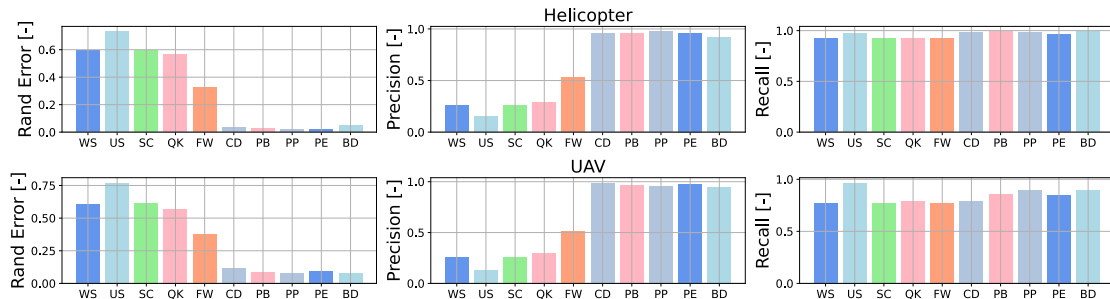
The results are shown in figure 7.10 for both the helicopter and the UAV data. First, we find that as the number of keyframes increases, the accuracy also increases. This comes with increased computational costs compared to the use of a single reference image. However, such costs can be justified since the system does not require real-time performance. Second, we observe that random selection of the keyframes can lead to a decrease in the mAP metric because the keyframes are selected from the output of the probabilistic object detector. Inaccurate annotations as sources for image matching can deteriorate the performance (also characterized by a high error bar in a single source image). In contrast, our uncertainty-driven mechanism improves accuracy by selecting only the bounding boxes that the detector is confident about. LA outperformed the baseline in this case. Overall, the results show that our design reduces the number of annotations required in the classification of tufts.

### 7.5.3. Results on Tuft Segmentation

For tuft segmentation, we evaluate our approach against several baselines (from classical image processing methods to deep learning) and perform ablations on loss functions.

To do so, we annotated 1000 segmentation masks for evaluation only. In the training steps, we use the curve detection (Steger, 1998) on 1000 tuft patches. The batch size of eight was used with a learning rate of 0.0002. For baselines, Felzenszwalb (Felzenszwalb & Huttenlocher, 2004) (FW), Quick (Vedaldi & Soatto, 2008) (QK), Watershed (Neubert & Protzel, 2014) (WS), Slic (Achant et al., 2012) (SC), unsupervised (Kanezaki, 2018)

<sup>2</sup>We note that more key-frames can be chosen from the available sets of annotations, at the cost of more manual efforts. However, our focus is on reducing such efforts to improve the applicability of our concept.



**Figure 7.11.:** Rand (Arganda-Carreras et al., 2015), precision and recall are reported for the ten baselines. The lower the better for the rand error, the higher the better for the precision and recall. Favorable results are observed for our approaches (CD, PB, PP, PE, and BD) over four alternatives.

	Completeness	Correctness	Quality	Point distance
<i>Helicopter In-Flight Images</i>				
CD	0.890±0.002	0.832±0.002	0.761±0.003	0.208±0.002
BD	<b>0.970±0.001</b>	0.881±0.009	0.828±0.007	0.113±0.002
PB	0.963±0.001	0.889±0.003	0.840±0.002	<b>0.085±0.001</b>
PE	0.930±0.004	0.881±0.002	0.840±0.001	0.176±0.002
PP	0.957±0.003	<b>0.889±0.004</b>	<b>0.841±0.005</b>	<b>0.085±0.002</b>
<i>Stratospheric UAV In-Flight Images</i>				
CD	0.262±0.001	0.810±0.002	0.619±0.003	1.348±0.002
BD	<b>0.846±0.005</b>	0.893±0.004	<b>0.775±0.001</b>	0.185±0.002
PB	0.801±0.003	0.898±0.010	0.760±0.004	0.132±0.003
PE	0.799±0.004	0.894±0.001	0.756±0.004	0.137±0.001
PP	0.809±0.005	<b>0.902±0.010</b>	0.762±0.007	<b>0.127±0.002</b>

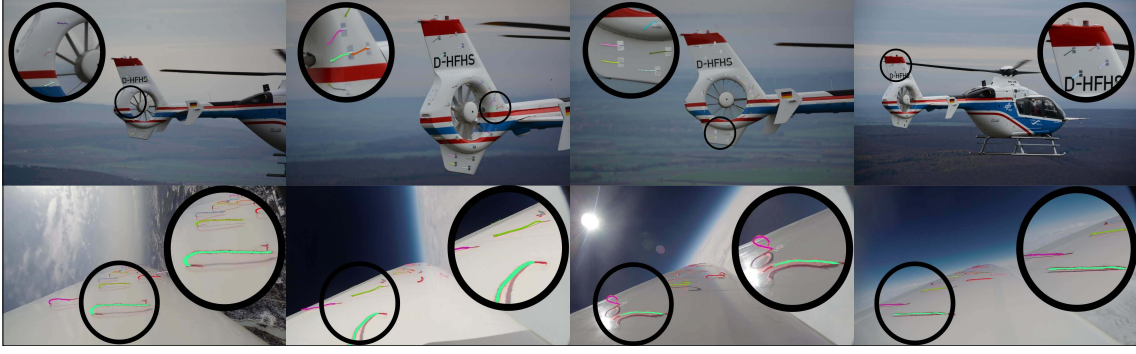
**Table 7.1.:** Segmentation evaluation results. The lower the better for point distance. Higher the better for other metrics.

(US) and Curve Detection (Steger, 1998) (CD) are chosen. These are the baselines that do not require segmentation masks, and there are open-source implementations for generic segmentation tasks. In addition to precision and recall, the adapted random error is used as an evaluation metric. Regarding the ablations on AL, we include PBCE loss only (PB), with a point distance (PP), with the edges (PE), and binary cross entropy with dice loss (BD). For the metric, instead of the pixel-wise intersect of union, we adapt a center-line quality metric, which is (a) completeness – a measure of the percentage of true prediction to all the labels, (b) correctness – a measure of the percentage of true prediction to all other predictions, and (c) quality – a measure on the percentage of true prediction to all predictions and the labels. Moreover, we also use the point distance in pixels as another metric to evaluate how well the algorithms are able to predict the start and end point of the tufts. We explain these measures on the project website.

The results are depicted in figure 7.11. We observe that for the rand error (Arganda-Carreras et al., 2015), the considered baselines perform poorly when compared to the ablation models. The same is observed for the precision, while the recall is only slightly lower than that of the ablation models. The best model in terms of the rand error and the precision metrics is PP, which is the proposed method from this chapter. In the recall metric, PB outperformed PP. The ablation results are reported in table 7.1. Again, we observe a significant improvement in all the metrics when compared to CD. From the experiments, we, however, find that no single loss can be uniformly superior when projected to four success criteria. One clear observation is that the proposed learning method improves over the simple curve detector, thereby validating the proposed approach.

#### 7.5.4. Results on Final Performance

Lastly, we evaluate the final performance. Two supervised instance segmentation networks, namely Mask RCNN and Cascaded-mask RCNN, are used as baselines. The implementations were based on the OpenMMLab toolbox. For training, the default OpenMMLab hyperparameters are used, except that the number of epochs had to be increased to 300, and the image dimensions were not resized to deal with small objects. Both for training and evaluation, we annotated 50 images each per dataset. We note that these annotations are costly, i.e., approximately 2200 tufts had to be annotated with semantic segmentation masks. Five training-test splits of ratio 60-40 were used.



**Figure 7.12.:** Final results are illustrated for the helicopter (top) and the UAV (bottom). The relevant portion of the images is zoomed in. Colored overlays indicate the segmentation masks per class. Our method can perform semantic segmentation tasks under severe perspective changes (such as in helicopter data), foreground changes of tufts, and lighting conditions. Model-based technique of instance semantic segmentation alone may fail as shown in figure 7.11.

	Mask RCNN	Cascaded RCNN	Ours
Helicopter	23.422±2.1822	11.198±2.4028	<b>60.729±2.1894</b>
UAV	34.032±1.2945	18.254±0.5844	<b>58.343±2.4532</b>

**Table 7.2.:** Quantitative results in mAP. Higher the better.

The results are reported in table 7.2. We observe that the supervised segmentation networks perform poorly. We attribute this to the lack of available training data. When no abundant training data is available, the results indicate that directly employing the end-to-end supervised deep learning techniques may pose challenges. On the other hand, with probabilistic approaches, we show how semantic segmentation can be performed without requiring any manual annotations of semantic segmentation masks. For qualitative results, figure 7.12 and the accompanying video demonstrate the overall performance, where we visually show that many of the tufts can be segmented with the correct semantics.

## 7.6. Summary

This chapter presented a learning system to provide automatic evaluations of in-flight images with tuft for flow visualization. For the first time to our knowledge, we developed an automatic tuft recognition system for flow visualization of aerial systems during real test flights. To achieve this, we performed data collection using two real application scenarios, namely a manned full-sized helicopter and a UAV flying in the stratosphere. A key challenge has been the lack of annotated data for supervised training. Using probabilistic approaches, we show how the annotation efforts can be reduced significantly. The experimental results demonstrate how the developed approaches can address the identified challenges.



---

## Realizing Telepresence Robots with Aerial Manipulation Capabilities

---

### 8.1. Motivation

Every critical infrastructure needs to be inspected and maintained. Often, these jobs are required within areas that are difficult to reach. Some concrete examples are inspecting oil and gas pipes in refineries, maintenance of antennas at cell towers, and repairing the blades of wind turbines. For such tasks, the current state-of-the-art is to deploy human technicians, who reach the high altitude areas through specialized equipment such as ropes, scaffolding, and cranes, and perform the required manipulation tasks. The current solutions are dangerous and costly due to inherent risks in ensuring the safety of technicians. One of the research directions for these applications is aerial manipulation ([Ollero et al., 2022](#)). An aerial manipulation system is composed of robotic manipulators and a controlled flying platform ([Bodie et al., 2020](#); [Kondak et al., 2014](#)). The platform enables coarse positioning, while the manipulator enables dexterous grasping and manipulation for complex tasks. Hence, these aerial platforms extend the mobility of robotic manipulators, which can be deployed at high altitudes above ground, increasing safety for human workers while reducing costs. Examples of aerial manipulation applications range from load transportation ([Bernard & Kondak, 2009](#)), contact-based inspection and maintenance in chemical plants ([Trujillo et al., 2019](#)), bridges ([Sanchez-Cuevas et al., 2019](#)), power line maintenance ([Cacace et al., 2021](#)), to sensor installations in forests for fire prevention ([Hamaza et al., 2019](#)).

In this chapter, the real world applications of aerial manipulators are envisioned for several industrial scenarios in dynamic and unstructured environments. For these industrial applications of aerial manipulators, our current interests are in the bilateral teleoperation concepts, i.e., a human operator remotely controls the robotic manipulator from a safe area on the ground and receives visual and haptic feedback from the robot. This increases human operator safety while the robots execute their tasks in dangerous environments ([Hulin et al., 2021](#); [Hirzinger et al., 2003](#)). Such a concept is motivated by having a robotic system with a human-in-the-loop, where the system can leverage human intelligence to reliably accomplish its missions. To realize this, existing works have focused on relevant components of the system, namely force feedback teleoperation under time delays ([Balachandran et al., 2021b](#); [Artigas et al., 2016](#)), shared autonomy ([Masone et al., 2018](#)), human-machine interfaces ([Kim & Oh, 2021](#); [Yashin et al., 2019](#); [Wu et al., 2018](#)), and robotic perception for aerial



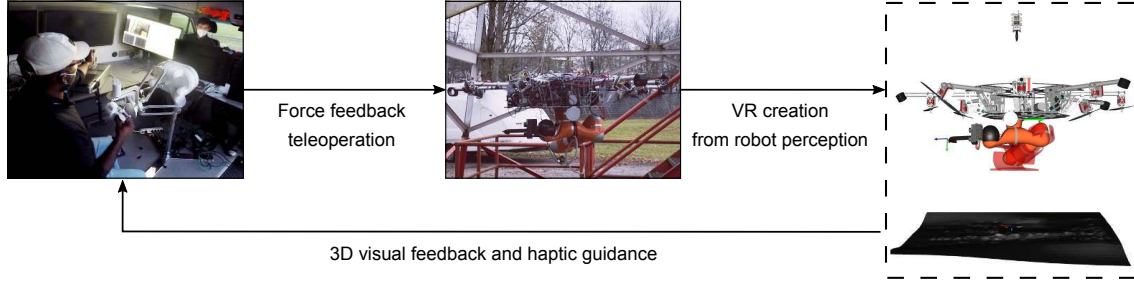
**Figure 8.1.:** Left: the cable-Suspended Aerial Manipulator, dubbed SAM (Sarkisov et al., 2019) during field experiment. Right: a ground station where an operator remotely controls the robotic arm through a haptic interface. In real world applications of bilateral teleoperation, the operator is often remotely located without visual contact to the robot.

manipulators (Karrer et al., 2016; Pumarola et al., 2019).

Building upon the aforementioned developments, we propose a virtual reality (VR)-based telepresence system for an aerial manipulation system. Figures 8.1 and 8.2 illustrate the main idea. The proposed system is intended for real world scenarios, where the remotely located robot performs aerial manipulation tasks, while its human operator is inside a ground station without having direct visual contact with the robot (figure 8.1). To this end, we propose a system which does not only involve a haptic device to enable the *sense of touch* for the operator, but also a VR to increase the *sense of vision* (figure 8.2). Although live video streams can also provide a certain level of situation awareness to the operator, several studies confirm that adding a virtual environment where one can change its sight-of-view, zoom in and out, and further provide haptic guidance supports the operator in performing the tasks (Pace et al., 2021; Whitney et al., 2020; Huang et al., 2019). Our own field studies also confirm that augmenting live video streams with 3D visual feedback and haptic guidance can enhance manipulation capabilities of aerial robots.

The main novelty of our VR-based concept is its realization with a *fully on-board perception* system for a *floating-base robot*, which *does not rely* on any external sensors like Vicon, or any pre-generated maps in *outdoor environments*. Instead, *multiple sensors*, namely LiDAR, a monocular camera, a pair of stereo cameras and inertial measurements units (IMUs) are jointly utilized (table 8.1). To achieve this, we propose object pose estimation and active learning pipelines. First, in order to virtually display industrial objects with known geometry, we provide a simple extension of a marker tracking algorithm (Wagner & Schmalstieg, 2007) by combining it with on-board Simultaneous Localization And Mapping (SLAM). Second, if the objects of interest are geometrically unknown, we devise a LiDAR-based pose estimation pipeline that combines LiDAR Odometry And Mapping (LOAM Zhang & Singh (2017)) with a pose graph, a point cloud registration algorithm (Besl & McKay, 1992), and a deep neural network (DNN)-based object detector (Lin et al., 2017). For both cases, the combinations are facilitated by an introspection module that identifies the reliability of the pose estimation. Finally, we present an active learning pipeline, which uses an explicit representation of DNN’s uncertainty, to generate the most informative samples for a DNN to learn from. This enhances the sample efficiency of deploying DNNs in outdoor environments. We identify certain real-world challenges and describe in detail





**Figure 8.2.:** In our system, the robot creates VR of its workspaces as a 3D visual feedback to the human operator, and further provides a haptic guidance. The main novelty of this work is the development of such a VR-based telepresence system for real-world applications.

how these introspective approaches can mitigate these challenges.

With the DLR SAM platform (Sarkisov et al., 2019), the feasibility and benefits of the proposed idea are implemented and closely examined. To this end, we first present ablation studies on the designed pipelines with indoor and outdoor datasets from robot sensors. Here, the influence of each component is examined with regard to mitigating the identified challenges, and we show the feasibility of creating real-time VR, which can closely match the real workspaces of the robot. Moreover, the effectiveness of the proposed method is shown through outdoor experiments within the industrial scenario considered. This scenario, which was designed under the scope of EU project AEROARMS (Ollero et al., 2018), is relevant to inspection and maintenance applications for the gas and oil industry. It involves pick-and-place and force-exertion tasks during the mission, which is to deploy a robotic crawler for automating pipe inspection routines. In addition, the SAM platform that performs peg-in-hole tasks with a margin of error of less than 2.5 mm is considered, which is one of the standard manipulation tasks in industrial settings. By executing over 70 executions of the aforementioned tasks over days and nights, from spring to winter, and with different users and locations, the benefits of our VR-based telepresence concept are illustrated for enhancing aerial manipulation capabilities in real-world industrial applications.

**Contributions and major claims.** First, we propose an advanced VR-based telepresence system for aerial manipulation, which provides a 3D visual feedback and a haptic guidance. The system neither requires any external sensors nor pre-generated maps, has been evaluated outside laboratory settings, and can cope with the challenges of a floating-base system. Second, we devise object pose estimation and active learning pipelines to realize the system in dynamic and unstructured environments. Challenges to existing methods are reported and several ablation studies are provided to validate our methods. This work suggests the relevance of robotic introspection in realizing VR-based telepresence system for aerial manipulation. Lastly, we perform exhaustive flight experiments over extended durations including 40 task executions in outdoor environments, 27 task executions within a user validation study, and the operation of the system at night. Thus, we establish the proposed concept as a viable future option for real-world industrial applications.<sup>1</sup>

<sup>1</sup>Link to the project website: <https://sites.google.com/view/vr-sam/>.

	Outside the laboratory settings?	No external sensors or pre generated map?	Floating-base manipulation system?	Multiple exteroceptive sensors?
Yashin et al. (2019)	✗	✗	✓	✗
Pohl et al. (2020)	✗	✓	✗	✗
Kohn et al. (2018)	✗	✓	✗	✗
Kim & Oh (2021)	✗	✗	✓	✗
Vempati et al. (2019)	✗	✗	✓	✗
Liu & Shen (2020)	✗	✓	✗	✗
Puljiz et al. (2020)	✗	✓	✗	✗
Ponomareva et al. (2021)	✗	✓	✗	✗
Ours	✓	✓	✓	✓

Table 8.1.: Comparisons between the existing VR-based robotic systems.

## 8.2. Related Work

The proposed VR-based concept advances the area of VR interfaces for robotics. The comparison of this concept to existing works is summarized in table 8.1. The current literature from different domains of robotic research is discussed, which are pose estimation (section 8.4.1 and 8.4.2) and active learning with DNNs (section 8.4.3). Importantly, we stress that this work is not to advance the state-of-the-art methods in these two areas. Rather, the aim is to apply and extend them to realize a working system for the given industrial scenarios. For example, the provided extension of a marker tracking algorithm with visual-inertial SLAM is not the main contribution of this work.

**Virtual Reality interfaces.** In the past, several VR interfaces have been widely utilized in robotics including aerial systems (Wonsick & Padir, 2020). So far, the presented approaches often create the VR either by using external sensors such as Vicon and a-priori generated maps. Notably, Vempati et al. (2019) utilizes a-priori generated maps for the applications of VR in aerial painting. For aerial manipulation, Yashin et al. (2019) uses the Vicon system to create the VR while Kim & Oh (2021) renders the environment with a portable sensor kit (Oh et al., 2017). Recently, many VR systems have gained interest for robotic manipulation. Many works (Haidu & Beetz, 2021; Zhang et al., 2020c) let a human perform demonstrations in VR and transfer the demonstrated manipulation skills to real robots. These works greatly show the synergy between VR and robotics. As we demonstrate the feasibility of creating VR with on-board sensors only, the work can contribute to many of these works in showing how one can create a VR for robotics.

On the contrary, many researchers aimed to provide VR of the remote scene by applying 3D reconstruction techniques (Ni et al., 2017; Kohn et al., 2018). For example, Kohn et al. (2018) presents an approach using an RGB-D camera. As the main challenge of reconstruction-based methods is the limited bandwidth in communication, Kohn et al. (2018) proposes an object recognition pipeline, i.e., replacing the detected object with sparse virtual meshes and discarding the dense sensor data. Pohl et al. (2020) uses an RGB-D sensor to construct a VR for affordance-based manipulation with a humanoid, while Liu & Shen (2020) and Puljiz et al. (2020) create augmented reality for a drone and a manipulator, respectively. Pace et al. (2021) conducts a user study and argues that the point clouds of RGB-D sensors are noisy and inaccurate (with artifacts), which motivates point cloud pre-processing methods for telepresence applications (Pace et al., 2021). In contrast, our approach is based on scene graphs (section 8.3.2) with pose estimation, which



is an alternative to 3D reconstruction methods. Finally, the main novelties are illustrated in table 8.1, which are the realizations of a VR-based telepresence system for outdoor environments using multiple sensors jointly. No external sensors or pre-generated maps are used, while dealing with specific challenges of a floating-base manipulation system, i.e., the surface that holds a robotic arm is constantly changing over time, thereby inducing motions for the attached sensors.

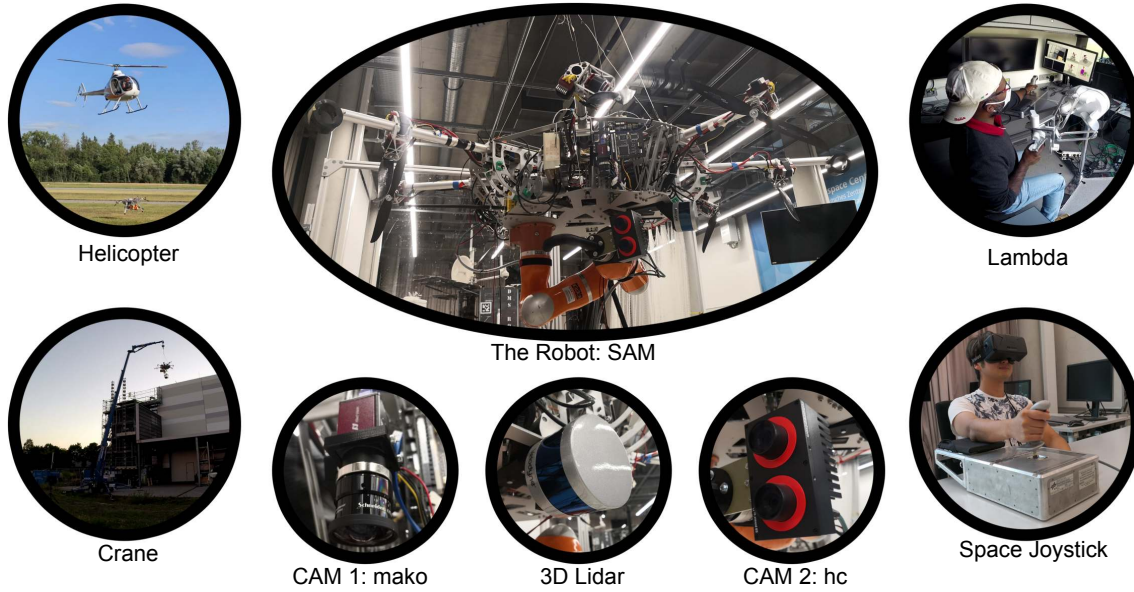
**Object pose estimation.** One of the crucial components in the proposed framework is object pose estimation algorithms. This is because we utilize a scene graph representation, which requires the 6D pose of the objects for creating a 3D display, as opposed to a 3D reconstruction of the remote site. As the literature is vast, we refer to the survey (He et al., 2021) for a comprehensive review. In this work, the main novelty is the working solutions for the considered application, which are tailored towards realizing the proposed VR system. For this, the two scenarios are discussed below. These are visual object pose estimation for objects of known geometry, and a LiDAR-based method for unknown geometry.

If the object is known and accessible a-priori, one of the robust solutions is to use fiducial marker systems. Fiducial markers, which create artificial features on the scene for pose estimation, are widely used in robotics. The use-cases are for creating the ground truths (Wang & Olson, 2016), where environments are known (Malyuta et al., 2020), for simplifying the problem in lieu of sophisticated perception (Laiacker et al., 2016), and also calibration and mapping (Nissler et al., 2018). However, as the herein aim is on real-time VR creation, this use-case demands stringent requirements on their limitations in run-time, inherent time delays, and robustness. Therefore, an extension of ARToolKitPlus is provided (Wagner & Schmalstieg, 2007) with an on-board visual-inertial SLAM system.

For LiDAR, point cloud registration is often used for pose estimation. By finding the transformation between the current scans and a CAD model of an object, we can obtain the 6D pose of an object. Broadly, point cloud registration algorithms can be classified as local (Park et al., 2017; Rusinkiewicz & Levoy, 2001; Besl & McKay, 1992) or global (Zhou et al., 2016), and model-based (Pomerleau et al., 2015) or learning-based (Wang & Solomon, 2019; Zhang et al., 2020b). As CAD models of objects are often not available in the given industrial scenario, a DNN-based detector and the idea of LOAM with pose graphs are combined in order to obtain robust object pose estimates that cope with occlusions, moving parts, and viewpoint variations in the scene.

**Active learning.** Field robotic applications of object detectors motivate active learning. Here, the need for labeled data can cause overhead in development processes, especially while considering a long-term deployment of learning systems in outdoor environments. For example, weather conditions can change depending on seasons, and we need to efficiently create labeled data. Active learning provides a principled way to reduce manual annotations by explicitly picking data that is worth being labeled. One way to autonomously generate the “worth” of an unlabeled sample is to use the uncertainty of DNNs. In the past, for robot perception, we find active learning frameworks using random forests, Gaussian processes, etc. (Narr et al., 2016; Mund et al., 2015) while for DNNs, MacKay (1992b) pioneered an active learning approach based on Bayesian Neural Networks, i.e., a probabilistic or stochastic DNN (Gawlikowski et al., 2023), which offers a principled method for uncertainty quantification. Recent works can also be found on active learning for DNN-based object detectors (Choi et al., 2021; Aghdam et al., 2019), where the focus is on adaptations of active learning to existing object detection frameworks. These include new acquisition functions (or selection criteria) and how uncertainty estimates are generated.

For uncertainty in DNNs, so-called Monte-Carlo dropout (MC-dropout Gal & Ghahramani



**Figure 8.3.:** The concept of SAM with its integrated sensors and human-machine interfaces.

(2016)) has gained popularity recently. The main advantage of MC-dropout is that it is relatively easy to use and scale to large datasets. However, MC-dropout requires a specific stochastic regularization called dropout (Srivastava et al., 2014). This limits its use on already well-trained architectures, because the current DNN-based object detectors are often trained with other regularization techniques such as batch normalization (Ioffe & Szegedy, 2015). Deep ensemble (Lakshminarayanan et al., 2017) is another scalable framework with a relaxed assumption on the model. Unfortunately, deep ensemble requires training of several large DNN models to form an ensemble. This technique is popular generally, but it is difficult to be utilized in active learning due to the inefficiency in training. In this chapter, a previous work (Lee et al., 2020b) on uncertainty quantification of DNNs is instead utilized. The main motivations are the scalability to large architectures and datasets, training-free feature that needs no changes in network architectures and no re-training, and the ability to model every layer of DNNs as Bayesian. These aspects can make the given framework well suited for active learning in practice, and thus, this work attempts to provide an extension to active learning for its real-world applications in robotics.

### 8.3. System Description, Problem Statement and Challenges

This chapter investigates how a robot can create a VR of a remote scene using on-board sensors and computations. This is to enhance the situational awareness of the human operator in real-world applications. To set the scene for the work, we first describe the system integrations that are needed to implement the proposed VR-based telepresence concept. Then, the problem of VR creation, using on-board sensors with a scene graph approach, is formulated. The limitations of the off-the-shelf methods are then presented, which hinder the realization of the proposed system in outdoor environments.

### 8.3.1. System Description

This section describes the used robotic systems with a focus on robot hardware, haptic devices, VR interfaces, and sensors. Main features of the system are also discussed. Figure 8.3 depicts an overview of our physical hardware.

**Robot hardware.** DLR’s SAM (Sarkisov et al., 2019) is a novel aerial manipulation system for inspection and maintenance applications. SAM is composed of three modules, namely a carrier, a cable-suspended platform, and a seven degrees of freedom (DoF) industrial robotic arm - KUKA LWR (Albu-Schäffer et al., 2007). The purpose of the carrier is to transport the manipulation system to a desired location. We use a crane in this work, which provides safety, versatility, robustness, and applicability for the considered industrial scenario <sup>2</sup>. Then, a platform attached to the carrier via a rope autonomously damps out the disturbances induced by the carrier, the environment, and the manipulator. This oscillation damping control is performed using eight propellers and three winches. Another important component of our system is the seven DoF torque-controlled KUKA LWR (Albu-Schäffer et al., 2007), which features significantly more powerful, versatile manipulation capabilities than many existing smaller manipulators. The main feature of the cable-suspension concept is that the weight of SAM is supported by the carrier. Thus, the required energy to carry an aerial robot arm can be reduced. This allowed us to scale down the overall size, from a helicopter-based system (Kondak et al., 2014) to a relatively smaller robot, which enables operation in confined spaces. The helicopter-based system had two rotors with an overall diameter of 3.7m, while SAM can fit within a 1.5m diameter. Moreover, the cables from the carrier can also be used to power SAM, which gives theoretically unlimited operation time. Sarkisov et al. (2019) can be referenced for the design and control of SAM in detail.

**Haptic devices and virtual reality interfaces.** In this work, two haptic devices, namely a space qualified haptic device called the Space Joystick RJo (Artigas et al., 2016), and also a six Dof force feedback device, Lambda (Force Dimension), are integrated in order to teleoperate the LWR on SAM. This work’s VR interface is based on Instant Player (Thomas et al., 2012), which is a lightweight software that runs on standard laptops without GPUs (enhancing portability). Instant Player also supports various hierarchies of a scene graph to create the required display. Facebook’s head mounted display Oculus is also integrated as an option and use Ubiquiti Bullet for the WiFi connection. The robot is equipped with advanced control strategies, namely whole body teleoperation, and adaptive shared control. The time-domain passivity approach of Artigas et al. (2016) is employed to obtain stable teleoperation control under communication time delays, packet loss and jitters. These control methods advance aerial manipulation capabilities. Coelho et al. (2021) and Balachandran et al. (2021a) present these concepts in more detail. The former presents a passivity-based framework to enable time-delayed teleoperation of different hierarchically-sorted tasks through the use of multiple input devices. Balachandran et al. (2021a) present a method to stabilize on-line adaptation of control authorities for the operator and the virtual assistance system in haptic shared control.

**Sensor choices and integration.** We integrate several sensors for measuring the robot’s own states as well as to perceive the environment. More specifically, a KUKA LWR (Albu-Schäffer et al., 2007) is equipped with torque and position sensors, which measure its joint torques and angles. Furthermore, we integrate other sensors on SAM for the

<sup>2</sup>On the other hand, there is no free lunch. Cranes may not be able to reach all the desired location as they require available access routes by ground. There are also several industrial tasks where smaller robotic arms with less DoF may be sufficient.

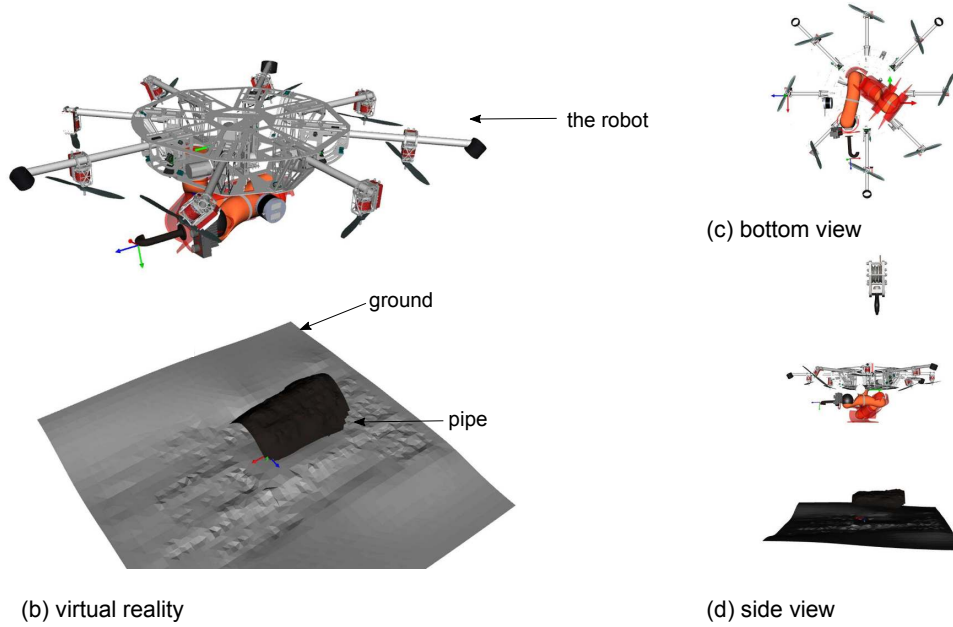
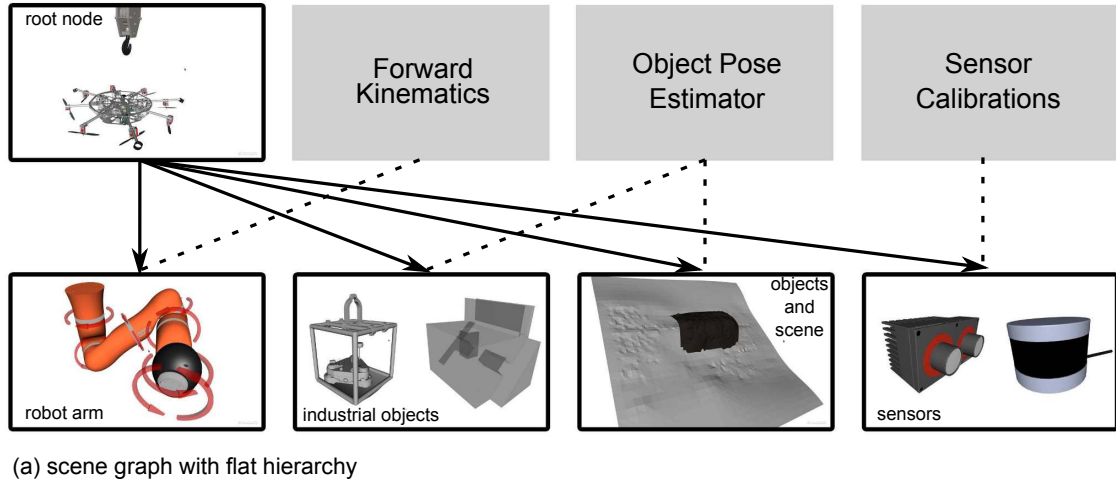
perception tasks. Firstly, a camera (the Allied Vision: mako) is integrated on the frame of SAM to stream the overall operational space of the robotic arm. This is because the operator prefers an eye-to-hand view, which is more natural for a human. The camera provides color images of 1292 by 964 px at 30Hz. Secondly, a stereo camera is integrated near the tool-center-point (tcp) of the robotic arm. This eye-in-hand set-up avoids occlusion of the camera view by the robotic arm and ensures proximity to the considered objects. These are crucial for the success of our image processing algorithms, i.e., the accuracy of visual marker tracking depends on the size of the markers and their distance to the sensors, while the depth sensing from the stereo depends on the baseline. We use a commercial 3D vision sensor, the Roboception Rcvissard, that provides built-in visual-inertial SLAM. The SLAM system originates from (Schmid et al., 2014; Lutz et al., 2020) but we refer the reader to the company for more details. Rcvissard streams 1280 by 960 px images at 25Hz and SLAM estimates can be acquired at 200Hz by fusing it with an IMU. Lastly, as a step towards the industrial application of SAM, we mount a Velodyne PUK-LITE LiDAR on the frame of SAM, which provides 3D point clouds of the scene at 10Hz. We intend to use LiDAR for 3D object pose estimation as well as navigation of SAM in outdoor environments. Note that the minimum range is set to 0.9m while the maximum range of 100m is utilized. We designed and integrated the sensor stacks so that the close range perception is not affected. All the perception algorithms are executed on the NVIDIA Jetson TX2.

### 8.3.2. Problem Formulation and Identified Challenges

Assume that SAM is performing manipulation tasks far away from the human operator. So, the operator does not have direct visual contact with the scene, and the robot has to enhance the situational awareness of the operator. For this, SAM creates a VR of its environment and workspaces using on-board sensing and computations, and further provides haptic guidance via virtual fixtures (Rosenberg, 1993). Followed by the system-level requirements, the problem formulation and the challenges are introduced (see figures 8.4 and 8.5).

The system level requirements are as follows. Firstly, the created VR has to accurately match the real remote site in real-time. This is because the operator needs visual feedback that reflects reality, and the performance of haptic guidance depends on the positioning accuracy and run-time. The latter is due to potential movements of the robot while hovering. Second, the robustness of the created VR is crucial to give a sense of trust to the human operator and further provide reliable haptic guidance. This means that the abnormalities in the object pose estimators are to be coped with, which often arise in outdoor environments. Last, the algorithms must run on-board the robot, and only send the transformation matrices through the WiFi network (apart from an initialization phase, where surface reconstructed 3D models are sent). This avoids overloading of the communication channel for teleoperation. For example, both the sparse LiDAR point clouds and the dense stereo point clouds must be processed first, and only the poses must be sent through the WiFi network. The pose information contains only six float values, while continuous streaming of the point clouds requires more memory that grows with the number of points.

To address the aforementioned requirements, this work relies on a scene graph approach (shown in figure 8.4). A scene graph is a general data structure with graph or tree-like representations. It is used by the VR/AR software (Thomas et al., 2012), in order to produce the real-time 3D visualizations. Mathematically, let  $S$  be a scene graph. It constitutes sets of nodes and edges, denoted by  $(V, E)$ . The nodes  $V$  are any 3D models, while the edges  $E$  represent the spatial relationships. The root node  $V_{\text{root}}$  is chosen to be the robot's base frame, which is a fixed coordinate of the SAM platform. Then, a flat hierarchy of the scene

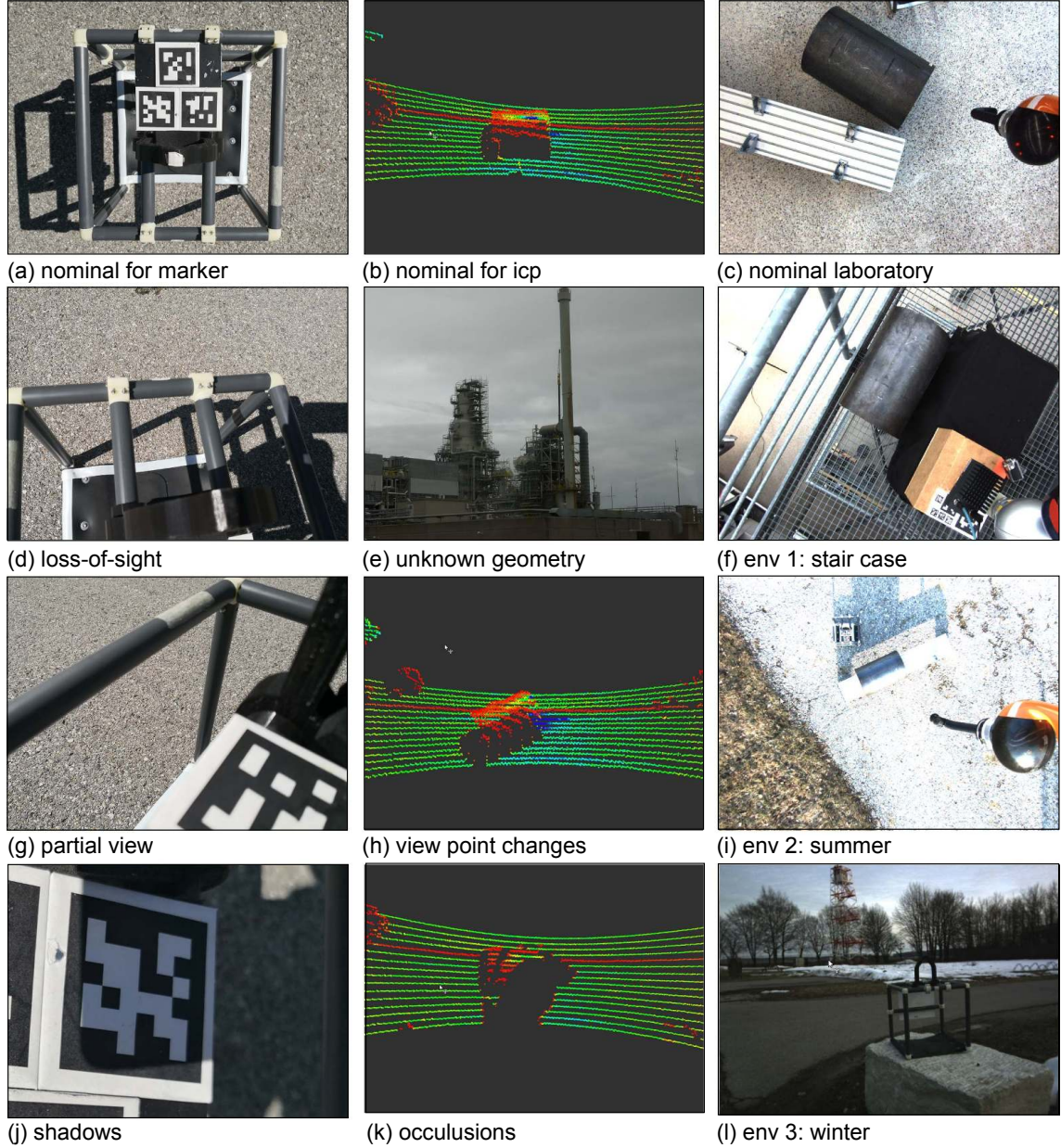


**Figure 8.4.:** The scene graph representation for the proposed VR framework. The root node is the base frame of the robot, while robot arm, industrial objects, scenes, and sensors are object nodes with transformation matrices as the edges. Forward kinematic provides state of the robot arm, and the fixed transformations to the robot sensors are obtained from extrinsic sensor calibrations. Then, the 6D estimates of object pose are obtained online using robot perception. Therefore, we focus on the object pose estimation for the development of the proposed system.

graph (Thomas et al., 2012) is assumed. This means the root node is a single “parent” to all other “child” nodes. Here, the models to be displayed in VR are the sets of industrial objects, reconstructed external scenes, robotic arm, and the robot sensors. For a node of the robotic arm  $V_{LBR}$ , the corresponding edges  $E_{LBR}^{root}$  are readily provided by the forward kinematics. Similarly, the edges of three sensors,  $E_{hc}^{root}$ ,  $E_{mako}^{root}$ , and  $E_{LiDAR}^{root}$ , are the outputs of the extrinsic camera calibration. These spatial relations or the relevant transformation matrices are therefore fixed for the sensor nodes  $V_{hc}^{root}$ ,  $V_{mako}^{root}$  and  $V_{LiDAR}^{root}$ .

On the contrary, the spatial relations of industrial objects and external scenes are





**Figure 8.5.:** Identified challenges for realizing our VR-based concept. Top: (a,d,g,j) show the challenges associated with a marker tracking algorithm. Middle: (b,e,h,k) depict the challenges associated with directly applying point cloud registration methods for pose estimation. In particular for (e), precise geometry of objects are not available for pipe inspection scenario as an example, and therefore, its CAD models must be reconstructed online. Bottom: (c,f,i,l) visualize different scenes that a learning-based method must cope with, when deployed for real world applications. For example, a DNN trained in a laboratory, may not generalize to the scenes with (f) stair cases.

constantly changing, leading to the problem of pose estimation. Here, we divide the problem formulation into two sub-problems. The first sub-problem is when the object is known a-priori with available 3D models ( $V_{o1}^{\text{root}}$ ,  $E_{o1}^{\text{root}}$ ), while the second sub-problem is when the object is semantically known a-priori, but no primitives on the geometry exist ( $V_{o2}^{\text{root}}$ ,  $E_{o2}^{\text{root}}$ ). For the former, the corresponding edges are to be estimated. The latter involves the estimation of both the nodes and the edges. As articulated in section 8.3.1, the

available raw sensor data are RGB camera images  $\mathbf{I} \in \mathbb{R}^{H \times W \times 3}$ , where  $H$  and  $W$  are the image height and width, respectively. The images are obtained either from the eye-in-hand stereo camera (denoted by  $hc$ ), or a monocular camera at the base (denoted by  $mako$ ). A LiDAR, which is located also at the base, generates scans that are represented by the point clouds  $\mathbf{P} = (\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_N) \in \mathbb{R}^{3 \times N}$ . We also have a visual-inertial SLAM system at the end effector of the robotic arm, which outputs the rotation matrix  $\mathbf{R}$  and translation vector  $\mathbf{t}$  between the coordinate frames of the camera and a fixed world frame. In summary, the problem of VR creation can be formulated as estimating  $\mathbf{E}_{o1}^{\text{root}}$ ,  $\mathbf{V}_{o2}^{\text{root}}$  and  $\mathbf{E}_{o2}^{\text{root}}$  using the available sensory data from different cameras, an IMU and a LiDAR.

For this problem, several existing approaches can be applied. However, several practical challenges of directly applying these approaches have been identified from the field work (depicted in figure 8.5). For objects of known geometry, we can resort to marker-based object pose estimation methods. For this, we cannot assume the holistic view of the markers. Violations of this assumption are caused by shadows, loss-of-sight, or partial views of the markers. This results in failures while using off-the-shelf marker tracking methods. Moreover, in an industrial scenario, we cannot assume the availability of precise CAD models. Thus, point cloud registration methods cannot be directly employed. Tracking is also subject to occlusions and moving objects in front of the LiDAR, e.g., the robotic arm, and significant viewpoint changes also result in less accurate 6D object poses while employing off-the-shelf methods such as the iterative closest point algorithm. Finally, while deploying data-driven approaches for field robotics, key to its success is the preparation of the data. The main challenges are the variations of scenes encountered during long-term deployment; darkness in the evenings or snow in winter are such examples. This means data has to be repeatedly collected for varying environmental conditions, which is a laborious process. So, the question is: how to make the data collection procedure more efficient so that DNNs can generalize. We address these challenges in the next section.

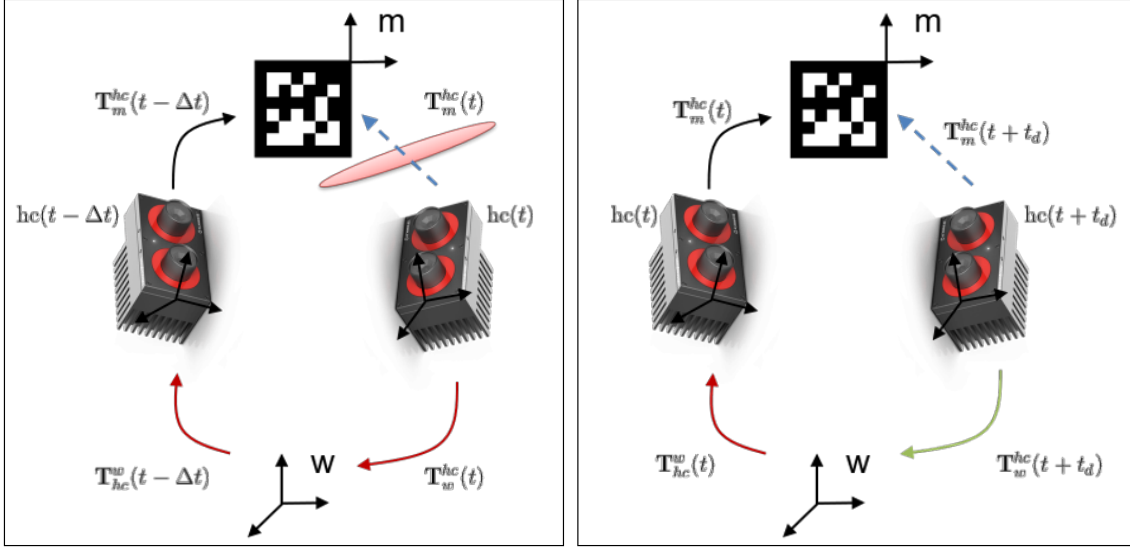
## 8.4. Introspective Perception Pipeline

The aim is to create a 3D display of the robot and the objects so that the human operator can remote control the robotic arm from a distance. If done in real-time, the operator can *see* the VR and perform the tasks. Haptic guidance via virtual fixtures can further help the human operator during the execution of challenging manipulation tasks. So far, we have formulated the problem and also outlined the practical challenges. As previously discussed, the scene graph creation problem relies on the accurate, fast, and reliable 6D object pose estimation algorithms for the industrial objects of known and unknown geometries. This section describes the proposed pipeline for the object pose estimation.

### 8.4.1. The Proposed Pipeline for Objects of Known Geometry

Once the objects to be actively manipulated are known a-priori, i.e., the CAD models are available and the objects are physically accessible, the fiducial marker systems (Wagner & Schmalstieg, 2007) can be exploited. These systems consist of a marker, which is a physical plane with black and white squared shapes (similar to QR codes), and a detection with a decoding algorithm. The key idea is to artificially create features on a plane that are physically attached to an object. Then, we can compute the pose of a camera in relation to a coordinate of the plane via a homography. Concretely, using the eye-in-hand ( $hc$ ) camera, the goal is to find the transformation matrix of the markers  $\mathbf{T}_m^{hc}$ , expressed in





**Figure 8.6.:** The proposed extension of ARToolKitPlus. Left: the position and orientation estimates of camera motion from a SLAM system infer the object when the markers are not detected. Right: linear and angular velocity estimates of the SLAM system are used with the time delay term  $t_d$  to predict the motion of the camera in  $t + t_d$  seconds.

the coordinate system of the camera, which consists of the rotation matrix  $\mathbf{R}_m^{hc}$  and the translation vector  $\mathbf{t}_m^{hc}$ . To do so, four corner points of the markers are extracted, which are expressed in the marker coordinates  $\mathbf{p}_m = (x_m, y_m, 0)^T$  (hence  $z_m = 0$  and given the size), and the image plane with pixels  $\mathbf{p}_{image} = (u_m, v_m, w_m)^T$ . Then, the optimizer:

$$\mathbf{h}(t) = \arg \min_{\mathbf{h}} \sum_{i=1}^4 \rho(\mathbf{p}_{i,image}(t), \mathbf{H}_{image}^m(t) \mathbf{p}_{i,m}(t)) \quad \text{where} \quad (8.1)$$

$$\mathbf{H}_{image}^m(t) = \left( \mathbf{R}_m^{hc}(t) + \frac{\mathbf{t}_m^{hc}(t)}{d} \mathbf{n}^T \right),$$

is the solution to the homography problem. Here,  $t$  denotes time,  $\mathbf{H}_{image}^m$  is the homography matrix with  $\mathbf{h}$  being its vector form, and  $\rho$  is a distance-based cost function. Knowing the homography matrix, the desired rotation and translation can be obtained given the parameters of the intrinsic camera calibration:  $d$  and  $\mathbf{n}$ . Typically, an algebraic formulation is used with the Direct Linear Transformation (DLT) algorithm (Andrew, 2001). We note that the fiducial marker systems are widely adopted as ground truths in the robotics community for their accuracy (Wang & Olson, 2016).

**Challenges.** However, many existing fiducial marker systems (Wagner & Schmalstieg, 2007; Wang & Olson, 2016; Malyuta et al., 2020; Laiacker et al., 2016) do not address this work’s application scenarios, where aerial manipulation tasks in outdoor environments are considered. For example, shadows that are created by the robot can often destroy certain shapes of the markers, and as a result, the methods would fail as the artificial visual features in the markers are occluded. Similarly, the eye-in-hand camera can lose the view of the marker as the manipulator and the base can move rapidly. Lastly, time delays that are inherent in these systems must be corrected in order to create a real-time virtual display of the scene. Next, the proposed solution to these challenges is described.

**Our solution.** To tackle these problems, we propose a robust marker localization pipeline (depicted in Algorithm 7) as an extension to ARtoolKitPlus (Wagner & Schmalstieg, 2007).

```

I      camera images from the eye-in-hand (hc) camera.
m      target marker identification.
input : i      identification numbers of additional markers  $i = 1, 2, \dots, n$ .
         $t_d$     time delay parameter, either online computed or prespecified.
         $\mathbf{T}_{hc}^w$  SLAM estimates of hc camera w.r.t a world coordinate.
output:  $\mathbf{T}_m^{hc}$  6D pose of the target marker m w.r.t the hc camera.

begin
  /* Initialization */
   $\mathbf{T}_m^{hc}(0), \mathbf{T}_i^{hc}(0) \leftarrow \text{multiART}+(\mathbf{I}) \ \forall i;$  // detect all the markers (equation 8.1)
   $\mathbf{T}_m^i \leftarrow \text{marker\_init}(\mathbf{T}_m^{hc}(0), \mathbf{T}_i^{hc}(0)) \ \forall i;$  // save all the relative poses
  /* Main Loop */
  while True do
     $\mathbf{T}_m^{hc}(t), \mathbf{T}_i^{hc}(t) \leftarrow \text{multiART}+(\mathbf{I}) \ \forall i;$  // detect the markers (equation 8.1)
    if all markers detected then
       $\mathbf{T}_{m,i}^{hc}(t), \mathbf{T}_{m,i}^{hc}(t) \leftarrow \text{trafo2m}(\mathbf{T}_m^{hc}(t), \mathbf{T}_i^{hc}(t), \mathbf{T}_m^i) \ \forall i;$  // transform to target
       $\mathbf{T}_m^{hc}(t) \leftarrow \text{ransac\_avg}(\mathbf{T}_m^{hc}(t), \mathbf{T}_{m,i}^{hc}(t)) \ \forall i;$  // ransac and average
       $\mathbf{T}_m^i \leftarrow \text{init\_update}(\mathbf{T}_m^{hc}(t), \mathbf{T}_i^{hc}(t)) \ \forall i;$  // update all the relative poses
    else if not all marker detected then
       $\mathbf{T}_{m,i}^{hc}(t), \mathbf{T}_{m,i}^{hc}(t) \leftarrow \text{trafo2m}(\mathbf{T}_m^{hc}(t), \mathbf{T}_i^{hc}(t), \mathbf{T}_m^i) \ \forall i;$  // transform to target
       $\mathbf{T}_m^{hc}(t) \leftarrow \text{ransac\_avg}(\mathbf{T}_m^{hc}(t), \mathbf{T}_{m,i}^{hc}(t)) \ \forall i;$  // ransac and average
    else if no marker detected then
       $\mathbf{T}_m^{hc}(t) \leftarrow \text{slam\_integrate}(\mathbf{T}_m^{hc}(t), \mathbf{T}_{hc}^w(t - \Delta t), \mathbf{T}_t^{hc}(t - \Delta t));$  // equation 8.2
       $\mathbf{T}_m^{hc}(t + t_d) \leftarrow \text{delay\_integrate}(\mathbf{T}_m^{hc}(t + t_d), \mathbf{T}_{hc}^w(t), \mathbf{T}_m^{hc}(t));$  // equation 8.3
    end
  end

```

**Algorithm 7:** Robust marker localization algorithm with Visual-Inertial SLAM

As an overview, the proposed pipeline utilizes multiple markers as well as the robots' SLAM system. To explain, multiple markers are placed on an object, where there exists a predefined target marker ID  $m$  and  $n$  additional markers with unique identifications, i.e.,  $i = 1, 2, \dots, n$ . This results in total,  $k = n + 1$  markers. At initialization, the algorithm detects all the markers, where *multiART+* is the function that executes a variant of the marker tracking method: ARtoolKitPlus (Wagner & Schmalstieg, 2007)). Using the eye-in-hand camera image  $\mathbf{I}$  (either the left or the right camera of the stereo setup), we obtain the initial 6D pose of the target marker  $\mathbf{T}_m^{hc}$  as well as all  $n$  additional markers  $\mathbf{T}_i^{hc}$  at  $t = 0$ . Then, we save the relative poses of all  $n$  markers to the target marker  $m$  (denoted  $\mathbf{T}_m^i$  for  $i = 1, 2, \dots, n$ ). This step is executed within the function *marker\_init*.

Then, the 6D pose of the target marker  $\mathbf{T}_m^{hc}$  is obtained in the main loop of Algorithm 7. The first step is to execute *multiART+*. Then, if all the  $k$  markers are detected, we transform the 6D pose of  $n$  additional markers to the target marker:  $\mathbf{T}_{m,i}^{hc} = \mathbf{T}_i^{hc} \mathbf{T}_m^i$  (executed with a function *trafo2m*). As this results in  $n$  additional 6D poses of the target marker  $m$ , we note them as  $\mathbf{T}_{m,i}^{hc}$  for  $i = 1, 2, \dots, n$ . Then, RANSAC (Fischler & Bolles, 1981) is applied to these estimates to remove the outliers, and then we perform averaging to reduce the variance (*ransac\_avg*). Then, the relative transformations  $\mathbf{T}_m^i$  are updated. If at least one marker is detected, the same step is applied to estimate the target marker without updating the relative transformations  $\mathbf{T}_m^i$ . We also note that RANSAC is skipped when less than three points are available. The described steps have two advantages. First, the accuracy and the orientation ambiguity of ARtoolKitPlus can be improved with RANSAC. Second, the algorithm is robust to loss-of-sight of the target marker, i.e., detecting only one of the markers is enough to still estimate the 6D pose of the target. Similar steps have been

presented in the past with several variants (Laiacker et al., 2016; Malyuta et al., 2020).

However, the algorithm must be robust to loss-of-sight on all the markers, and further compensate for the time delay. This is achieved by extending the algorithm with SLAM estimates. The overview is depicted in figure 8.6. As a first step, we propose to address the problem of complete loss-of-sight on all the markers by integrating SLAM estimates of camera motion with respect to its inertial coordinate, i.e., utilize the estimated transformation between the hc camera to a world coordinate of our SLAM system w:  $\mathbf{T}_{hc}^w(t)$ . If no markers are detected in the main loop of the algorithm, one can still estimate the target marker  $\mathbf{T}_m^{hc}(t)$  by integration (executed within the function *slam\_integrate*):

$$\mathbf{T}_m^{hc}(t) = \mathbf{T}_w^{hc}(t)\mathbf{T}_{hc}^w(t - \Delta t)\mathbf{T}_m^{hc}(t - \Delta t). \quad (8.2)$$

In equation 8.2,  $\mathbf{T}_w^{hc}(t)\mathbf{T}_{hc}^w(t - \Delta t)$  is a relative transformation of camera motion from time  $t-1$  to  $t$  and we assume a static object. In a similar fashion, the time delay of the system  $t_d$  can be computed (executed with a function *delay\_computation*) and corrected with the SLAM algorithm by kinematics:

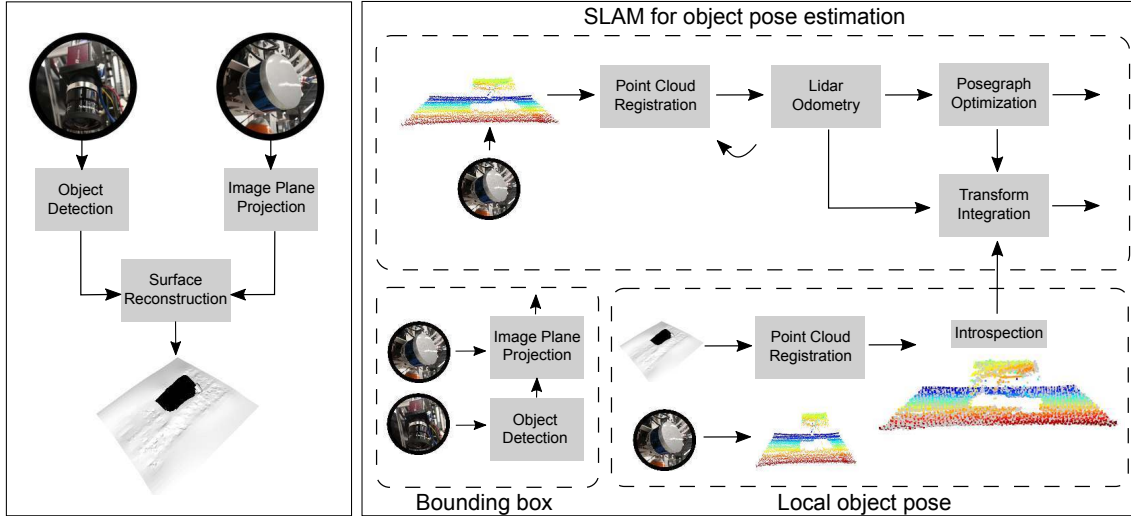
$$\mathbf{T}_m^{hc}(t + t_d) = \mathbf{T}_w^{hc}(t + t_d)\mathbf{T}_{hc}^w(t)\mathbf{T}_m^{hc}(t), \quad (8.3)$$

which is executed within a function *delay\_integrate*. Note that the time delay is present in any perception system (e.g., rectifying an image), fiducial marker systems as well as the communication delays. In equation 8.3,  $\mathbf{T}_{hc}^w(t)$  and  $\mathbf{T}_t^{hc}(t)$  are computed using SLAM and multi-marker tracking. On the other hand,  $\mathbf{T}_w^{hc}(t + t_d)$  can be computed using linear and angular velocity estimates of SLAM, multiplied by the delay time  $t_d$ . These two steps have several advantages. The algorithm is robust to the found failure modes of fiducial marker systems as it copes with missing marker detection, and time delays are incorporated by using velocity signals and computed delay time. Furthermore, maximum run-time of the algorithm can be pushed up to 200Hz, which is the rate of visual-inertial SLAM estimates. The algorithm deals also with drifts of SLAM estimates by using relative motion estimates only when the marker detection is lost. Note that the proposed method is simple but can be an effective way of exploiting the commodity vision sensors with SLAM modules in order to improve the robustness of the fiducial marker systems.

#### 8.4.2. The Proposed Pipeline for Objects of Unknown Geometry

Whenever we cannot assume the availability of the markers, the 6D pose of the objects can be estimated using depth sensors such as a LiDAR with the point cloud registration methods. For example, within the intended industrial application, the markers cannot be used for estimating the pose of the pipe. This is because in oil and gas refineries, the pipes are often very long while their inspection points are generally unknown a-priori. Concretely, given the incoming streams of point clouds  $\mathbf{P}(t)$  and the point clouds of the object  $\mathbf{Q}$  from a CAD model, the goal is to find the rotation and translation between  $\mathbf{P}(t)$  and  $\mathbf{Q}$ . Here, the point clouds  $\mathbf{P}(t)$  contain the points  $\mathbf{p}_i(t) \forall i$  with their coordinates lying at the weighted centroid  $\mathbf{c}_p$ . Similarly, the point clouds  $\mathbf{Q}$  contain the points  $\mathbf{q}_i(t) \forall i$  with their coordinate systems defined at the weighted centroid  $\mathbf{c}_q$ . Defining  $\mathbf{c}_q$  to be aligned with the coordinate system of the LiDAR  $l$ , the 6D pose of an object  $o$  can be obtained by matching the two point clouds:  $\mathbf{p}_i = \mathbf{R}_o^l \mathbf{q}_i + \mathbf{t}_o^l$ . This goal of finding  $\mathbf{R}_o^l$  and  $\mathbf{t}_o^l$  is often formulated as,

$$\mathbf{R}_o^l, \mathbf{t}_o^l = \arg \min_{\mathbf{R}_o^l, \mathbf{t}_o^l} \sum_i \rho(\|\mathbf{p}_i - \mathbf{R}_o^l \mathbf{q}_i - \mathbf{t}_o^l\|), \quad (8.4)$$



**Figure 8.7.:** The proposed LiDAR pose estimation pipeline. (a) Initialization. Using an object detector, the point clouds that belong to the objects of interest can be obtained for a surface reconstruction technique. (b) Main loop. The object poses are estimated by combining a SLAM technique with local object poses from a point cloud registration method. An object detector computes the regions of occlusions and dynamic objects for masking out.

where  $\rho$  is again a distance-based cost function, e.g. typically a mean squared error.

Commonly, the solution is obtained by first computing the rotation, and then the translation. Centering all the points  $\mathbf{p}_i$  and  $\mathbf{q}_i$  at their respective origins:

$$\bar{\mathbf{p}}_i = \mathbf{p}_i - \mathbf{c}_p \quad \text{and} \quad \bar{\mathbf{q}}_i = \mathbf{q}_i - \mathbf{c}_q \quad \text{such that} \quad \mathbf{R}_o^l = \arg \min_{\mathbf{R}_o^l} \sum_i \rho(\|\bar{\mathbf{p}}_i - \mathbf{R}_o^l \bar{\mathbf{q}}_i\|), \quad (8.5)$$

the goal is to find the rotation matrix that aligns the centered point clouds. Defining the correlation matrix as  $\mathbf{C} = \sum_i \bar{\mathbf{p}}_i \bar{\mathbf{q}}_i^T$  and its singular value decomposition as  $\mathbf{C} = \mathbf{U} \mathbf{\Lambda} \mathbf{V}^T$ , the rotation can be estimated by the orthogonal Procrustes algorithm, while the translation can be obtained from the weighted centroids  $\mathbf{c}_p$  and  $\mathbf{c}_q$  after rotation:

$$\mathbf{R}_o^l = \mathbf{U} \mathbf{V}^T \quad \text{and then} \quad \mathbf{t}_o^l = \mathbf{c}_q - \mathbf{R}_o^l \mathbf{c}_p. \quad (8.6)$$

This assumes the correspondences between each point to be known. In practice, however, the correspondences are often not known, and the Iterative Closest Point (ICP) algorithm is often used (Park et al., 2017; Rusinkiewicz & Levoy, 2001; Besl & McKay, 1992). Intuitively, the ICP algorithm iterates the following steps: (a) finding the closest point in the transformed point cloud for each point:  $\min \rho(\mathbf{P}, \mathbf{Q})$ , (b) estimating the transformation using equation 8.6, and (c) applying the found transformation to all points and iterating all the steps until a convergence criterion is reached. As the ICP algorithm is subject to local minima, ICP is often initialized by employing the global registration methods such as Zhou et al. (2016) or using higher-level features at the first step of the ICP algorithm.

**Challenges.** Unfortunately, such a strategy does not fully address the current use-case for point cloud registration. This is because of the aerial manipulation tasks for an inspection and maintenance scenario. We outline the resulting failure modes in figure 8.5. First, the strategy assumes the availability of a precisely known object geometry  $\mathbf{Q}$  from CAD models. Unfortunately, this assumption is invalid in our industrial scenario, as the CAD model of the objects that belong to external environments is unknown, e.g. CAD

model of oil or gas pipe. Even though the refineries may have a 3D geometry of the site, there exist erosion and other changes to their initial model. Second, a holistic view of the object cannot be assumed. In the set-up, robotic arms and other objects can occlude the object of interest, resulting in partial and overlapping views of the point cloud. Lastly, as we deal with a floating base system where the base of a robotic arm is not fixed, viewpoint challenges can occur. This challenges the out-of-box point cloud registration methods for LiDAR systems in equations 8.4, 8.5 and 8.6, which contain sparse point clouds.

**Our solution.** To this end, we propose a 6D object pose estimator using a LiDAR. The pipeline is depicted in figure 8.7. The proposed algorithm consists of an initialization step and a multi-process main loop. At initialization, the CAD model of the scene is reconstructed online by exploiting an object detector. In the main loop, three parallel processes are created: a bounding box estimator that computes the locations of the occluding and moving objects to be masked out, a SLAM pipeline that computes the object poses in a global reference frame, and a local object pose estimator that estimates the object poses locally. The combination of local and global methods is to mitigate the challenges related to the non-holistic view of the object. The SLAM estimates can deal with perspective changes by matching the scans sequentially, but suffer from drift. The local method, whenever it is reliable, can be exploited to reduce the drift of the SLAM system. Lastly, what motivates the multi-process architecture is the efficiency, i.e., LiDAR odometry can run at a faster rate than the others that can be executed only at a slower rate.

First, the pipeline is initialized by creating the CAD model of the object online. This is because one cannot always assume a known geometry of the object in the targeted applications, i.e., the target point cloud  $\mathbf{Q}$  is not available, and consequently its CAD model  $\mathcal{O}$  for the VR. Yet, from the specification of the given task, e.g. pick and place an inspection robot on a pipe, what we know a-priori is the semantics of the objects of interest, e.g. a pipe. Therefore, one can still create the CAD model of the object  $\mathcal{O}$  online by finding the point clouds that belong to the objects of interest  $\mathbf{P}_o(0) \in \mathbf{P}(0)$  and applying a surface reconstruction technique once. For this, we train a DNN-based object detector (Redmon et al., 2016) using the eye-to-hand camera (mako). Defining this DNN as a parametric function  $f_\theta$  with its input as an image  $\mathbf{I}$ , the goal of a 2D object detector is to classify and locate the objects in an image; for the object semantics  $c$ , e.g.  $c \in \{\text{pipe, robotic arm, cage}\}$ , the classification probability  $p_c$  (a score between 0 and 1), and the location as a bounding box  $\mathbf{b}_c \in \mathbb{R}^4$  in the given image, the 2D object detector returns the tuples:

$$\{c, p_c, \mathbf{b}_c\} = f_\theta(\mathbf{I}) \quad \text{with} \quad \mathbf{b}_c = [u_{c,1} \quad v_{c,1} \quad u_{c,2} \quad v_{c,2}]^T. \quad (8.7)$$

Here, the bounding box is described by two points in the image with the heights ( $h = u_{c,1}$  and  $h = u_{c,2}$ ) and the widths ( $w = v_{c,1}$  and  $w = v_{c,2}$ ) which are the top left and the bottom right corner of the box that contains the object  $c$ . Further defining the target object  $c = o$  and using the extrinsic calibration parameter between the LiDAR and the eye-to-hand camera  $\mathbf{T}_{mako}^l$  to transform all the point clouds  $\mathbf{P}(0)$  to the image plane, we can find the point clouds that belong to  $o$ :

$$\mathbf{P}_o(0) = [\mathbf{p}_j = (x_j, y_j, z_j)] \quad \text{such that} \quad j = \{i \mid u_{o,1} \leq x_{i,image} \leq u_{o,2}, v_{o,1} \leq y_{i,image} \leq v_{o,2} \forall i\}. \quad (8.8)$$

This means that all the LiDAR scans  $\mathbf{p}_i(0)$  to the image plane are transformed, which results in  $\mathbf{p}_{i,image} = (x_{i,image}, y_{i,image})$ . Then, we obtain the indices  $j$  of the point clouds that lie inside the bounding box of our target object  $\mathbf{b}_o$  and crop the original point cloud  $\mathbf{P}(0)$  to obtain the point clouds  $\mathbf{P}_o(0)$  that only contain the information about our target

object. Applying a surface reconstruction technique (Kazhdan et al., 2006), the CAD model of the target object  $\mathcal{O}$  can be created. In this way, we can still create the VR of the scenes with the objects of unknown geometry.

Next, the main loop of our algorithm is described. The first process is the bounding box estimator. This process tackles the problem of occluding and moving objects  $c = u$  by estimating the bounding box of other objects  $u$  in the LiDAR coordinate system, which is for actively removing the points that belong to the occluding and moving objects  $u$ . Similar to before, the object detector and the extrinsic calibration can be used to obtain:

$$\mathbf{P}_u(t) = [\mathbf{p}_u = (x_u, y_u, z_u)] \quad \text{s.t.} \quad u = \{i \mid u_{u,1} \leq x_{i,image} \leq u_{u,2}, v_{u,1} \leq y_{i,image} \leq v_{u,2} \forall i\}, \quad (8.9)$$

where  $\mathbf{P}_u(t)$  is the point cloud that belongs to the objects  $u$  at time  $t$ . Then, the bounding boxes of the objects can be computed in the xy-plane of the LiDAR coordinate system. Defining this as  $\mathbf{b}_u^l$ , and examining all the point clouds:

$$\mathbf{b}_u^l = [\min(x_u) \quad \min(y_u) \quad \max(x_u) \quad \max(y_u)]^T. \quad (8.10)$$

Note that the projection of all the point clouds to the image plane can be inefficient for embedded CPU computations. Therefore, a separate process is assigned. Also, moving averages and a margin are implemented. Lastly, these bounding boxes are used to mask out the occluding and moving objects in all other modules.

Then, in the main loop, a LiDAR-based SLAM (inspired by LOAM Zhang & Singh (2017)) is employed to address the problem of viewpoint changes. Again, a naive strategy is to perform point cloud registration between the reconstructed CAD model of the target object  $\mathcal{O}$  and the incoming point cloud scans  $\mathbf{P}$ . However, if the initially constructed object  $\mathcal{O}$  is significantly different from the current point cloud  $\mathbf{P}$ , the point cloud registration method may perform poorly due to less overlap between the two point clouds. Therefore, our key idea is to rely on a LiDAR odometry pipeline. The pipeline performs the registration between the consecutive point clouds and sets the coordinate of the constructed object  $\mathcal{O}$  as a global reference, which does not suffer from significant viewpoint changes. However, this approach suffers from drift, i.e., accumulation of errors. To address this, two mechanisms are introduced. The first mechanism is a pose graph optimization:

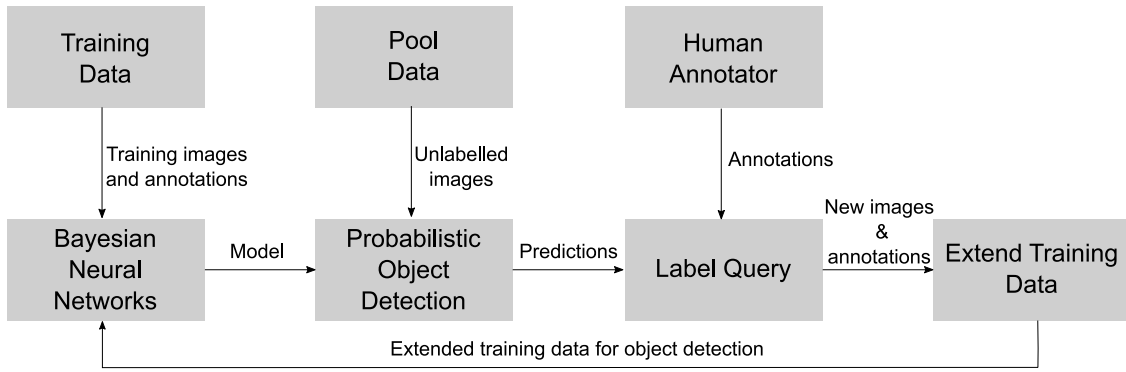
$$\{\mathbf{T}_i\} = \arg \min \lambda \sum_i \sum_{(\mathbf{P}, \mathbf{Q}) \in \mathcal{K}_i} \|\mathbf{T}_i \mathbf{P} - \mathbf{T}_{i+1} \mathbf{Q}\|^2 + \sum_{i < j} \sum_{(\mathbf{P}, \mathbf{Q}) \in \mathcal{K}_{ij}} \rho(\|\mathbf{T}_i \mathbf{P} - \mathbf{T}_j \mathbf{Q}\|), \quad (8.11)$$

where  $\lambda$  determines the weight of a cost between two consecutive scans within the keyframes, and  $\rho$  is a robust function, e.g. set to L2 norm in our case. Here, the framework of Choi et al. (2015) that performs robust pose graph optimization is applied, which is less prone to the errors of pairwise registration. Second, we propose to combine local object poses that are obtained by performing point cloud registration of incoming scans with the target object  $\mathcal{O}$ . Whenever the confidence estimates of the local object poses are high (or above a specified threshold), we reset the SLAM system with initialization from the local object pose estimator. In this way, we account for the drift of the SLAM system.

### 8.4.3. The Proposed Active Learning Framework

So far, the proposed object pose estimators are described. Here, our pipeline relied on a DNN-based object detector. This has been used for the online creation of a CAD model, and to rule out any occlusions and moving objects. As our entire system relies on a DNN





**Figure 8.8.:** Active learning for generating labeled data more efficiently. Instead of randomly selecting the images to be labeled, we query the most informative samples from Bayesian Neural Networks.

for the VR creation, we next propose an active learning framework to obtain the required performance in DNNs within the context of field robotics.

**Challenges.** The problem is on the training and the deployment of DNN-based systems for various environments including both indoor and outdoor conditions. The challenge lies in realizing a DNN-based system for long-term operations in outdoor environments. This is due to the necessity and the manual preparations of large amounts of high quality, annotated data which can cover the variety of the operational conditions. For example, as illustrated in figure 8.5, a DNN trained from the annotated images of the laboratory environments may not generalize to outdoor environments. Similarly, the seasonal variations of the scene from summer to winter can cause similar effects in the deterioration of the generalization performance. Therefore, each change in the scene may require an iterative process of collecting and annotating the data. As this can be a long and tedious process, we attempt to find a principled solution that guides the process of gathering the required data for the field deployments of the DNN-based systems.

**Our solution.** To this end, we now describe a pool-based active learning pipeline (Cohn et al., 1996). To explain, active learning is a class of machine learning paradigms, where labeled data is not available for a supervised learning problem. Instead of obtaining annotations for all available unlabeled data, active learning attempts to only label fewer but the most informative data. Intuitively, the aim of active learning is to create a learning system that chooses by itself what data it would like the user to label. As opposed to the heuristic choice of the user, active learning enables a DNN to select small amounts of data, guiding the user in the data creation process. In many applications, this means that a pool of unlabeled data needs to be collected by the robot first. As we use visual learning methods, a camera set-up could replace the deployment of the robot for data collection. Note that up to this stage, the procedure is similar to standard supervised learning settings in robotics. Then, instead of annotating all the available data manually, fewer images are then autonomously selected by the active learning algorithm. Deep learning models are trained from these fewer images and finally deployed to the robot.

Figure 8.8 illustrates the overall idea behind active learning. In a pool-based approach, a model is trained on an initial training set, which is often small. Then, the model selects a subset of data points from a pool of unlabeled data and asks a human to label the selected data points. The selection involves a decision-making process, which is performed through the choice of an acquisition function. Based on the updated training set, a new model is



trained. Repeating this process, we can reduce the amount of labeling required to train a learning system. We present such a system for DNNs, which relies on an uncertainty quantification technique for DNNs. These are namely Bayesian Neural Networks (BNNs) and probabilistic object detection. To explain, our algorithm 8 depicts the working principle overall. Using an initial training set  $\mathcal{D}_{\text{init}}$ , we train a BNN which is denoted as  $p(\boldsymbol{\theta}|\mathcal{D}_{\text{init}})$ . Then, for a user-specified number of query steps  $Q$ , we first select the most informative, top  $K$  samples from the pool of unlabeled images:  $\mathcal{D}_{\text{pool}}$ . This is achieved by estimating the uncertainty from BNNs (denoted  $p(\mathbf{y}_i^* | \mathbf{x}_i^*, \mathcal{D}_{\text{train}})$ ). Then, we label the selected images, and the BNN is updated with the new training set. For a more detailed explanation, we next present each of these components, namely the BNNs, the uncertainty of the BNN-based object detector, and query step through the acquisition function.

### Bayesian Neural Networks for Uncertainty Quantification

One of the crucial components of our active learning pipeline is BNNs. BNNs allow for the uncertainty quantification in DNN predictions. We note that our previous works on BNNs (Lee et al., 2020b; Humt et al., 2020) are being extended to the active learning framework for object detection. Chapter 4 can also be referenced. Our description below aims to provide the basic formulation within the context of its application to active learning.

For this, consider a supervised learning set-up with input-output pairs  $\mathcal{D} = \{\mathcal{X}, \mathcal{Y}\} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N$ , where  $\mathbf{x}_i \in \mathbb{R}^D$ ,  $\mathbf{y}_i \in \mathbb{R}^K$ . Similar to previous sections, we define a DNN as a parametrized function  $f_{\boldsymbol{\theta}} : \mathbb{R}^D \rightarrow \mathbb{R}^K$ , where  $\boldsymbol{\theta} \in \mathbb{R}^P$  is a vectorized form of all DNN weights or parameters, e.g. all the weights of the convolution kernel or the weights and biases of a multi-layer perceptron. In a standard DNN, we typically aim to minimize the loss function:  $\min_{\boldsymbol{\theta}} \frac{1}{|\mathcal{D}|} \sum_{(\mathbf{x}, \mathbf{y}) \in \mathcal{D}} \mathcal{L}(f_{\boldsymbol{\theta}}(\mathbf{x}), \mathbf{y}) + \frac{\delta}{2} \boldsymbol{\theta}^T \boldsymbol{\theta}$  where  $\delta$  is an  $L_2$ -regularizer, and  $\mathcal{B} \subset \mathcal{D}$  denotes mini-batches. The resulting solution is a single hypothesis of a local maximum-a-posteriori (MAP) solution  $\hat{\boldsymbol{\theta}}$ . To the contrary, BNNs explicitly express DNNs as probability distributions over DNN model parameters  $\boldsymbol{\theta}$  given the data  $p(\boldsymbol{\theta}|\mathcal{D})$ , which is also known as the posterior distribution over the DNN model parameters:

$$p(\boldsymbol{\theta}|\mathcal{D}) = \frac{p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathcal{D})}. \quad (8.12)$$

As an application of Bayes' theorem, where a prior distribution over the model parameters  $p(\boldsymbol{\theta})$  is specified, along with the likelihood  $p(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta})$  and the model evidence  $p(\mathbf{y}|\boldsymbol{\theta})$ . Once the posterior distribution over the weights is obtained, the prediction of an output for a new input datum  $\mathbf{x}^*$  can be obtained by marginalizing the likelihood  $p(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta})$  with the posterior distribution. This step of marginalization can be used for active learning:

$$p(\mathbf{y}^*|\mathbf{x}^*, \mathcal{D}) = \int p(\mathbf{y}^*|\mathbf{x}^*, \boldsymbol{\theta})p(\boldsymbol{\theta}|\mathcal{D})d\boldsymbol{\theta}. \quad (8.13)$$

This indicates that the uncertainty estimates for a DNN prediction  $\mathbf{y}^*$  can be obtained through combining different hypotheses of model parameters, resulting in the predictive distribution  $p(\mathbf{y}^*|\mathbf{x}^*, \mathcal{D})$ . Another implication of the formulation is the reliance on posterior probabilities  $p(\boldsymbol{\theta}|\mathcal{D})$  for uncertainty quantification. Chapter 2 explains marginalization.

Unfortunately, estimating the posterior is a challenging task, and has been one of the central topics in research of BNNs (Gawlikowski et al., 2023). While the reasons are myriad, one of the reasons is the lack of a closed-form solution due to the nonlinearities of DNNs that prohibit the validity of conjugate priors (Bishop & Nasrabadi, 2006). As a result,

the use of approximation techniques of Bayesian inference such as variational inference or Monte Carlo Markov Chain (MCMC) sampling has been researched with a focus on dealing with the high dimensionality of DNN weight-space and the scalability with respect to large amounts of data that DNNs typically assume. For the computations of the posterior, the proposed pipeline relies on the approaches of [Lee et al. \(2020b\)](#); [Hunt et al. \(2020\)](#). These works are well suited for active learning in robotics, due to the demonstrated scalability to large architectures and datasets ([Lee et al., 2020b](#)). The extension of [Lee et al. \(2020b\)](#) in the automation of the hyperparameter tuning via Bayesian Optimization ([Hunt et al., 2020](#)) can also be exploited in every query step of active learning.

### Uncertainty Estimation for Object Detectors

Having obtained the posterior probabilities of BNNs, the uncertainty estimates can now be computed for the underlying object detector. A key challenge is the adaptation of BNNs for the object detector that may rely on several post-processing steps ([Harakeh et al., 2020](#)). As we use anchor-based detectors such as Retinanet ([Lin et al., 2017](#)) (as these types of object detectors can provide real-time performance on the Jetson TX2 as opposed to region-proposal approaches or end-to-end pipelines), one needs to deal with miscorrespondence between the anchor predictions and final outputs, and (ii) hard cut-off behavior in the non-maximum suppression (NMS) step ([Lin et al., 2017](#)).

For these, the BayesOD framework ([Harakeh et al., 2020](#)) is employed, which infers the output distributions from the BNN’s predictive distributions. In BayesOD, the samples of the BNN’s predictive distributions are clustered in anchor level in order to derive the uncertainty estimates of the object detection. For this, one can assume that the clusters contain  $M$  number of anchors. We further assume the highest classification score as the center of this cluster (indexed by 1) and other anchors of the cluster are considered as measurements to provide information for the center (denoted as  $\hat{\mathbf{c}}_i$  and  $\hat{\mathbf{b}}_i$ ). Then, the uncertainty estimates for classification  $p_{[\hat{\mathbf{c}}_1, \dots, \hat{\mathbf{c}}_M]}(\mathbf{c}|\mathbf{x}^*, \mathcal{D}_{\text{train}})$  and regression  $p_{[\hat{\mathbf{b}}_1, \dots, \hat{\mathbf{b}}_M]}(\mathbf{b}|\mathbf{x}^*, \mathcal{D}_{\text{train}})$  are:

$$\begin{aligned} p_{[\hat{\mathbf{c}}_1, \dots, \hat{\mathbf{c}}_M]}(\mathbf{c}|\mathbf{x}^*, \mathcal{D}_{\text{train}}) &\propto p_{\hat{\mathbf{c}}_1}(\mathbf{c}|\mathbf{x}^*, \mathcal{D}_{\text{train}}) \prod_{i=2}^m p(\hat{\mathbf{c}}_i|\mathbf{c}, \mathbf{x}^*, \mathcal{D}_{\text{train}}), \\ p_{[\hat{\mathbf{b}}_1, \dots, \hat{\mathbf{b}}_M]}(\mathbf{b}|\mathbf{x}^*, \mathcal{D}_{\text{train}}) &\propto p_{\hat{\mathbf{b}}_1}(\mathbf{b}|\mathbf{x}^*, \mathcal{D}_{\text{train}}) \prod_{i=2}^m p(\hat{\mathbf{b}}_i|\mathbf{b}, \mathbf{x}^*, \mathcal{D}_{\text{train}}), \end{aligned} \quad (8.14)$$

where  $p_{\hat{\mathbf{c}}_1}(\mathbf{c}|\mathbf{x}^*, \mathcal{D}_{\text{train}})$  indicates the per-anchor predictive distribution of the cluster center and  $\prod_{i=2}^m p(\hat{\mathbf{b}}_i|\mathbf{b}, \mathbf{x}^*, \mathcal{D}_{\text{train}})$  is the likelihood. ([Harakeh et al., 2020](#)) can be referenced here.

### Acquisition Functions for Query Generation

Another component of active learning is the acquisition function which ranks the images in the pool set. The defined acquisition function typically uses the uncertainty estimates to evaluate how informative each image in the pool set is. In object detection, as there could be several object instances in an image, the information scores for each detected instance within an image are aggregated into one final score. Once such scores are obtained for all the images in the pool set, the top  $K$  images can be queried for human annotation, which is then stacked into the training set. The model is then retrained with the new and larger training set, and the process repeats. As the acquisition function is a selection mechanism of active learning, its design can influence the performance of the learning framework.

Within a BNN-based object detector, the uncertainty estimates can be obtained for both the classification and the bounding box regression. Hence, one of the design choices is

	$\mathcal{D}_{\text{init}}$	The initial annotated training data.
	$\mathcal{D}_{\text{pool}}$	The unlabeled data.
<b>input :</b>	$Q$	The number of query steps.
	$K$	The size of the query per step.
<b>output :</b>	$f_{\theta}$	The trained DNN-based object detector.
	$\mathcal{D}_{\text{train}}$	The annotated training data.

```

begin
  /* Initialization */
   $p(\theta|\mathcal{D}_{\text{init}}) \leftarrow \text{create\_BNN}(\mathcal{D}_{\text{init}});$  // Apply Lee et al. (2020b); Humt et al. (2020)
   $\mathcal{D}_{\text{train}} \leftarrow \text{update\_data}(\mathcal{D}_{\text{init}});$  // Initialize the training set from  $\mathcal{D}_{\text{init}}$ 
  /* Main Loop */
  for all the number of query steps  $Q$  do
     $p(y_i^* | \mathbf{x}_i^*, \mathcal{D}_{\text{train}}) \leftarrow \text{prob\_detector}(\mathcal{D}_{\text{pool}}) \forall i;$  // Evaluate uncertainty on a pool set (equations 8.14, 8.13)
     $\mathcal{D}_{\text{selected}} \leftarrow \text{query}(\mathcal{D}_{\text{pool}}, K);$  // Query from the pool set (equation 8.15)
     $\mathcal{D}_{\text{new}} \leftarrow \text{generate\_annotations}(\mathcal{D}_{\text{selected}});$  // The user or human supervisor annotates the images
     $\mathcal{D}_{\text{train}} \leftarrow \text{update\_data}(\mathcal{D}_{\text{new}});$  // Update the training set by adding new annotated data
     $p(\theta|\mathcal{D}_{\text{train}}) \leftarrow \text{create\_BNN}(\mathcal{D}_{\text{train}});$  // Apply Lee et al. (2020b); Humt et al. (2020)
  end
end

```

**Algorithm 8:** Deep Active Learning using Bayesian Neural Networks

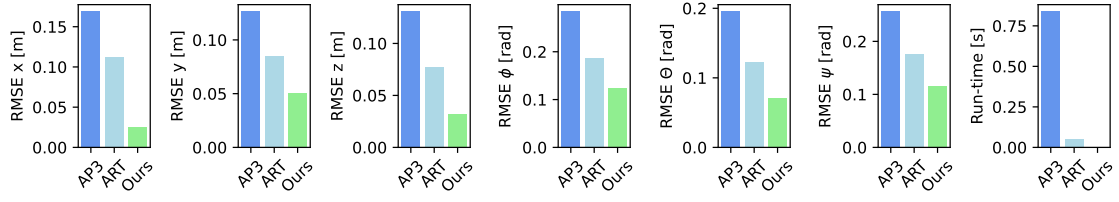
on how to effectively combine the two different types of uncertainty measures - one on semantic uncertainty, and the other on spatial uncertainty (Feng et al., 2022a). Defining the combination function  $\text{comb}(\cdot)$  as a weighted sum or max operation (Choi et al., 2021), and also the aggregation function  $\text{agg}(\cdot)$  as a sum or average operation (Roy et al., 2018):

$$\mathcal{A}(\mathbf{x}_k) = \text{agg}_{j \in N_k} (\text{comb}(\mathcal{U}_{j,cls}, \mathcal{U}_{j,reg})), \quad (8.15)$$

where  $\mathcal{U}_{j,cls}$  and  $\mathcal{U}_{j,reg}$  are the information scores of the  $j$ -th detection instance on an image, for the classification and the regression tasks respectively. A mechanism of this acquisition function is to first combine both the semantic and spatial uncertainty by either a weighted sum or maximum operation, and then sum or maximize over the combined score per detection instance. What motivates the given choice is the handling of the problem itself. The combination operation is to deal with having to combine the two different tasks per instance of an object detector, and the aggregation operation is to handle the multiple instances in a single image (Feng et al., 2022b). What remains are the information scores for both classification and regression tasks:  $\mathcal{U}_{j,cls}$  and  $\mathcal{U}_{j,reg}$  respectively. Then,

$$\mathcal{U}_{j,cls} = \sum_{i=1}^{|C|} \mathcal{H}(p(c_i|\mathbf{x}^*, \mathcal{D}_{\text{train}})) \quad \text{and} \quad \mathcal{U}_{j,reg} = \mathcal{H}(p(\mathbf{b}|\mathbf{x}^*, \mathcal{D}_{\text{train}})), \quad (8.16)$$

which rely on the Shannon Entropy measure  $\mathcal{H}(\cdot)$  - an indicator of how uncertain a distribution is. In the case of classification, we assume categorical distributions over the classes  $c_i$ , while we assume multivariate Gaussian distributions for the bounding box regression  $\mathbf{b}$ . Importantly, what motivates optimizing the given entropy measure is its equivalence to maximizing the information gain of a model (MacKay, 1992b).



**Figure 8.9.:** Root Mean Squared Error (RMSE) and run-time are reported for the baseline methods and the proposed extension to the ARToolKitPlus. Lower the better for both the measures.

## 8.5. Experiments and Evaluations

In this work, a VR-based telepresence system is proposed, which is to provide real-time 3D displays of the robots' workspace and also haptic guidance to a human operator. The main contribution is the realization of such a system using robotic perception and active learning methods. A key focus has been on introspection. This section therefore evaluates the proposed pipelines by examining how the created VR can match the real remote scenes, and if the identified challenges in figure 8.4 are addressed by the proposed pipelines. Then, the results from the field experiments are presented, in order to characterize the effectiveness of the overall system in advancing aerial manipulation for real-world applications.

### 8.5.1. Ablation Studies and Evaluations

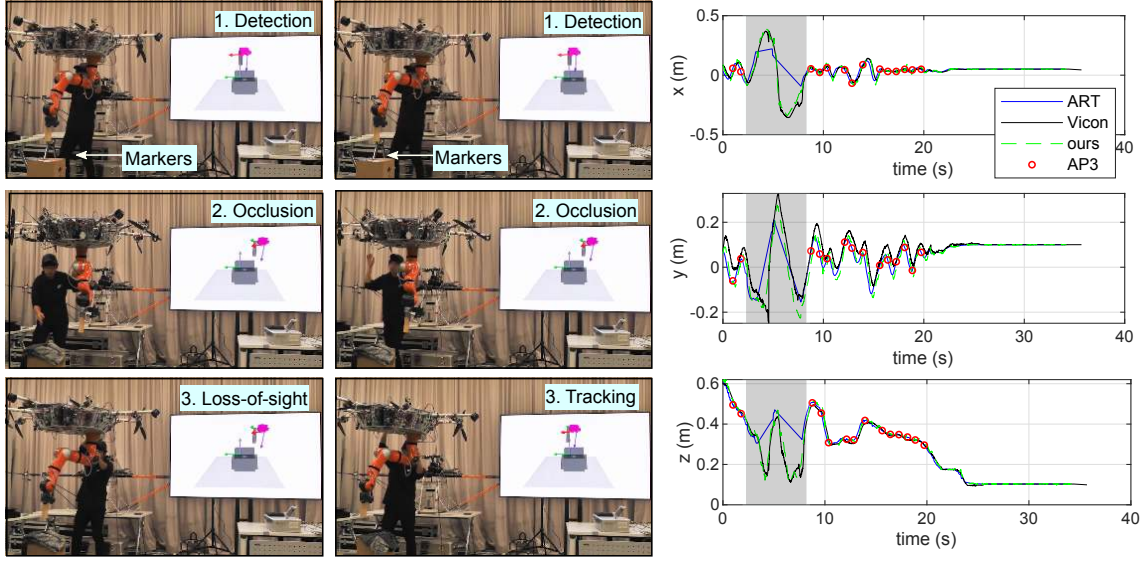
Several ablation studies are provided for the insights behind the presented algorithms. In particular, empirical evaluations of the devised algorithms, when facing the outlined challenges in figure 8.4, are the goals of this particular section.

#### Visual Object Pose Estimation for Known Objects

To recap, the marker tracking algorithms can be exploited for creating the VR with known objects. Here, the identified challenges are the shadows, the loss of sight, or the partial views of the markers, which can cause a mismatch between the real remote scene and the VR. To address these challenges, the SLAM estimates of commodity visual-inertial sensors have been integrated, and here, validation of the devised algorithm is performed. To this end, the accuracy, the run time, and the robustness of the proposed algorithm are examined.

**Experiment setup.** For this, the ground truth of the relative poses between the objects and the camera is measured using a Vicon tracking system. Then, the algorithms are evaluated on the sequences that simulate the peg-in-hole insertion tasks. The Vicon measurements represent the ground truth of the object poses for the indoor environments. To evaluate the effectiveness of the proposed algorithm against the identified challenges, the observed failure modes of the existing marker tracking systems are manually created. The baselines are the Apriltag3 (Wang & Olson, 2016) (denoted as AP3) and the ARToolKitPlus (Wagner & Schmalstieg, 2007) (denoted as ART), which represent plug-and-play alternatives. Particularly, as our algorithm extends the ARToolKitPlus with SLAM estimates, this choice of the baseline enables a direct comparison. Five repetitions are conducted in total.

**Results.** The results are reported in figures 8.9 and 8.10. In figure 8.10, the estimated trajectories of the relative poses are compared with the Vicon measurements. We observe that our pipeline is robust against the loss-of-sight problems with a hand-eye camera. On



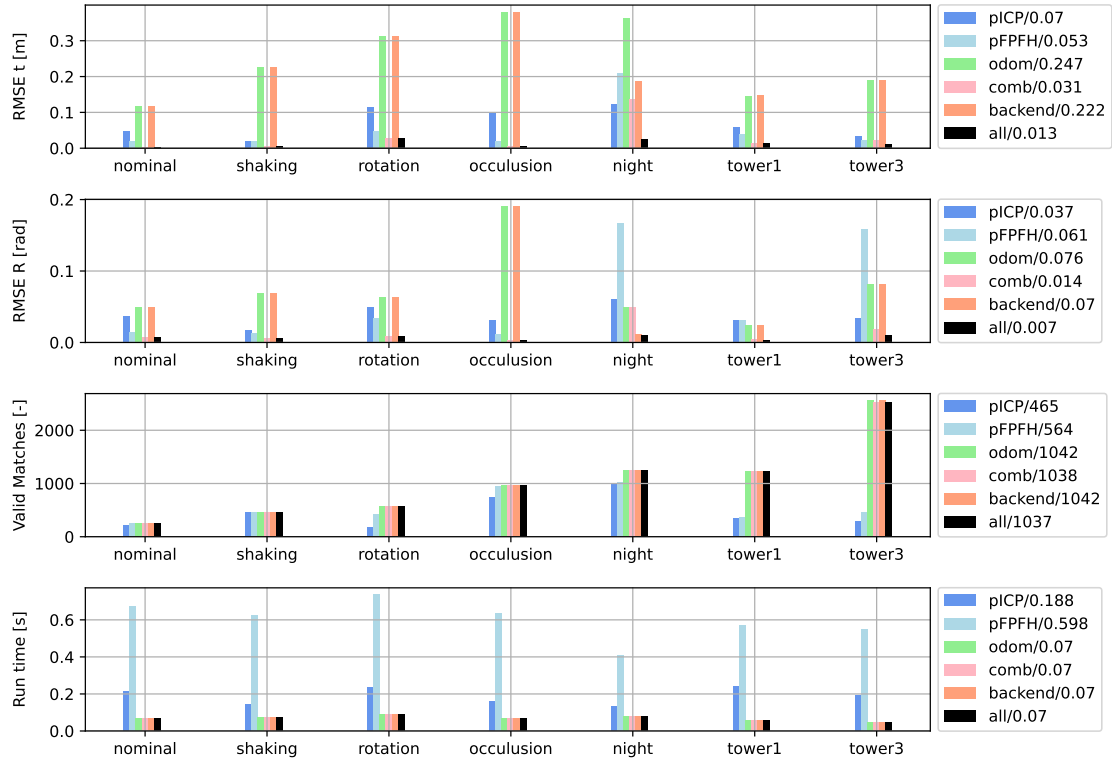
**Figure 8.10.:** Left: the existing marker tracking algorithms under the loss-of-sight of the markers are evaluated. Middle: the proposed extension of the marker tracking algorithm with SLAM estimates is evaluated under the same scenario of the loss-of-sight. Right: the estimated positions from the baselines and the proposed algorithm are compared. These results indicate that the proposed algorithm can cope with the loss-of-sight of the markers and the time-delays, thereby justifying the design choices of the algorithm. Three markers of size 2.5cm, a marker of 6.25cm and a marker of 10cm are used in this evaluation scenario.

the other hand, the alternatives, namely AP3 and ART, produce jumps as no markers are detected (between  $t=2$  to  $t=8$  as an example) when the camera loses sight of the markers. This is due to the design of the algorithm where the SLAM estimates of the camera pose are integrated out whenever the markers are not detected. Furthermore, ART suffers from a time delay, while AP3 has both a time delay and a slow run-time. Moreover, the proposed algorithm can compensate for the time delay, resulting in slightly more accurate estimates. The corresponding Root Mean Squared Errors (RMSE) are reported in figure 8.9 along with the run-time. We observe that AP3 is slow when using high-resolution images, and this results in more errors as the trajectories are compared. In the approach, these trajectories are relevant for creating the VR with object localization methods. In this experiment, the proposed method yields the least errors and fastest run-time. We attribute the former to the robustness against the loss-of-sight of the markers, while the latter is due to the integration of the inertial sensors. These analyses of the accuracy, the robustness, and the run-time validate the proposed algorithm. Moreover, the success of the algorithm is visually demonstrated in the video attachment, in addition to figure 8.10 (a-b).

### LiDAR Object Pose Estimation for Objects of Unknown Geometry

For the external scenes without the availability of a precise geometry, a LiDAR-based object pose estimation pipeline has been proposed. The features of the pipeline are to deal with the challenges that are outlined in figure 8.4. Thus, the aim now is to validate the components of the pipeline using the collected visual-inertial-LiDAR datasets.

**Experiment setup.** For this, the point clouds and the visual data are collected in various situations. Within the controlled lab environment the following scenarios are created:

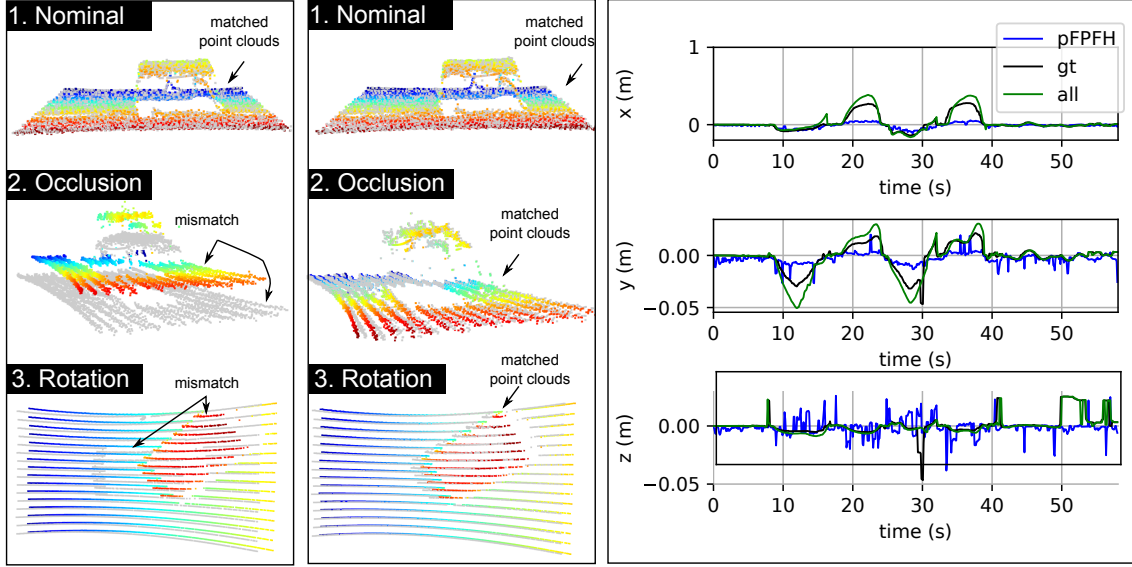


**Figure 8.11.:** Statistical analysis of the baselines and our pipeline. Lower the better for the RMSE and run-time, and higher the better for the number of valid point cloud matches.

a “nominal” scenario where the sensors ideally are pointed to an object statically, a “shaking” scenario in which imperfect hovering of the robot creates sensor movements, a “rotation” scenario where the robot rotates around the object, and a “occlusion” scenario in which the robot arm and other objects move to occlude the target object. These scenarios represent the identified challenges during a manipulation task (e.g. see figure 8.4). To further evaluate the proposed pipeline in a realistic use-case, additional scenarios are considered in outdoor environments. These are: a “night” scenario where the sensor data were collected during a manipulation task at night, a “tower 1” and “tower 2” scenarios where the data is similarly acquired at two different locations. Importantly, the given extensive evaluations over varying conditions are motivated by the considered industrial scenarios where this work aims to build a working system that goes beyond the proof-of-concept prototypes.

For the baselines, the off-the-shelf methods such as point-to-point ICP (Besl & McKay, 1992; Babin et al., 2019), point-to-plane ICP (Park et al., 2017; Rusinkiewicz & Levoy, 2001) and the combination with the global registration methods (Zhou et al., 2016) are compared. The pairwise registration is denoted pICP (with coarse-to-fine matching strategy) whereas pFPFH denotes the global registration method. These are to examine the vanilla object pose estimators without specific measures to address the identified challenges. For brevity, only the best performing ones between the point-to-point and the point-to-plane methods are reported. Furthermore, we compare our method without different components to evaluate the contributions of each module to the final performance. These are the pure odometry (odom), odometry with posegraph backend (backend), combination of odometry



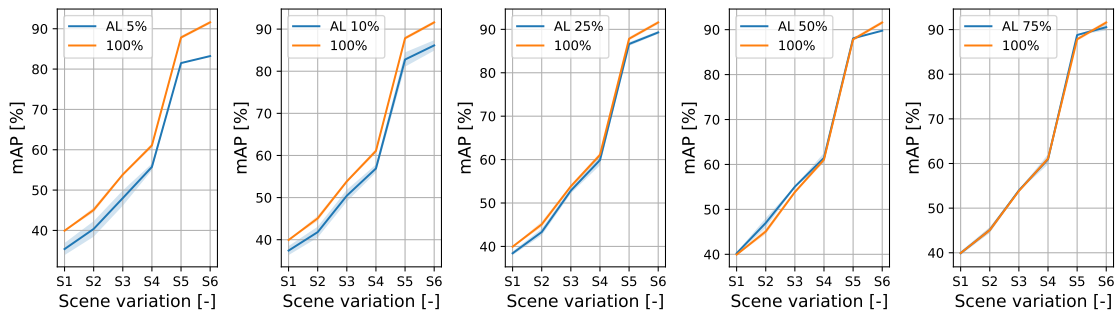


**Figure 8.12.:** Qualitative results for the LiDAR-based object pose estimators. (a) Evaluation of a vanilla object pose estimator. (b) Evaluation of the proposed object pose estimator. (c) Plots of the estimated positions from a baseline (pFPFH) and our approach (all) against the ground truth (gt) measurements.

and local object poses (comb) and the proposed object pose estimator (all). For better insights, we evaluate these methods with the masked out dynamic part of the scene while existing works motivate the importance of masking out the dynamic parts of the scene in the SLAM context. With these baselines, closely following [Babin et al. \(2019\)](#); [Park et al. \(2017\)](#), the RMSE of the translation (RMSE t) and the rotation (RMSE R), the number of valid matches, and the run-time of each algorithm are measured.

**Results.** The results are reported in figures 8.11 and 8.12. In these experiments, the proposed object pose estimator yielded the least RMSE for both translation and rotation. Odometry with and without the back-end suffers from drift over time, while the vanilla methods such as pICP and pFPFH perform poorly when the view point changes are significant. The latter is qualitatively shown in figure 8.12, while the number of matches in figure 8.11 also indicates their relatively poor registration between the target and the source point clouds. supports the claims of this work on the identified challenges. Moreover, as shown in figure 8.11, it can be seen that all the components introduced, namely pose-graph, local object poses, and odometry, contribute to the accuracy of the estimates. With respect to the run-time, the odometry-based methods are real-time capable, which we attribute to the significant overlap between two consecutive LiDAR scans that helps the ICP algorithm to converge faster. On the other hand, pFPFH is the slowest in terms of run-time because it relies on several components such as feature extraction, correspondence matching, and refinement through ICP. All these results are consistently observed across seven scenarios with varying degrees of severity. These experiments justify and validate the proposed methods and their design choices. Importantly, the key take-away is the effectiveness of combining object pose estimators with SLAM methods for floating-base systems, which can handle the failure cases of conventional object pose estimators via introspection.





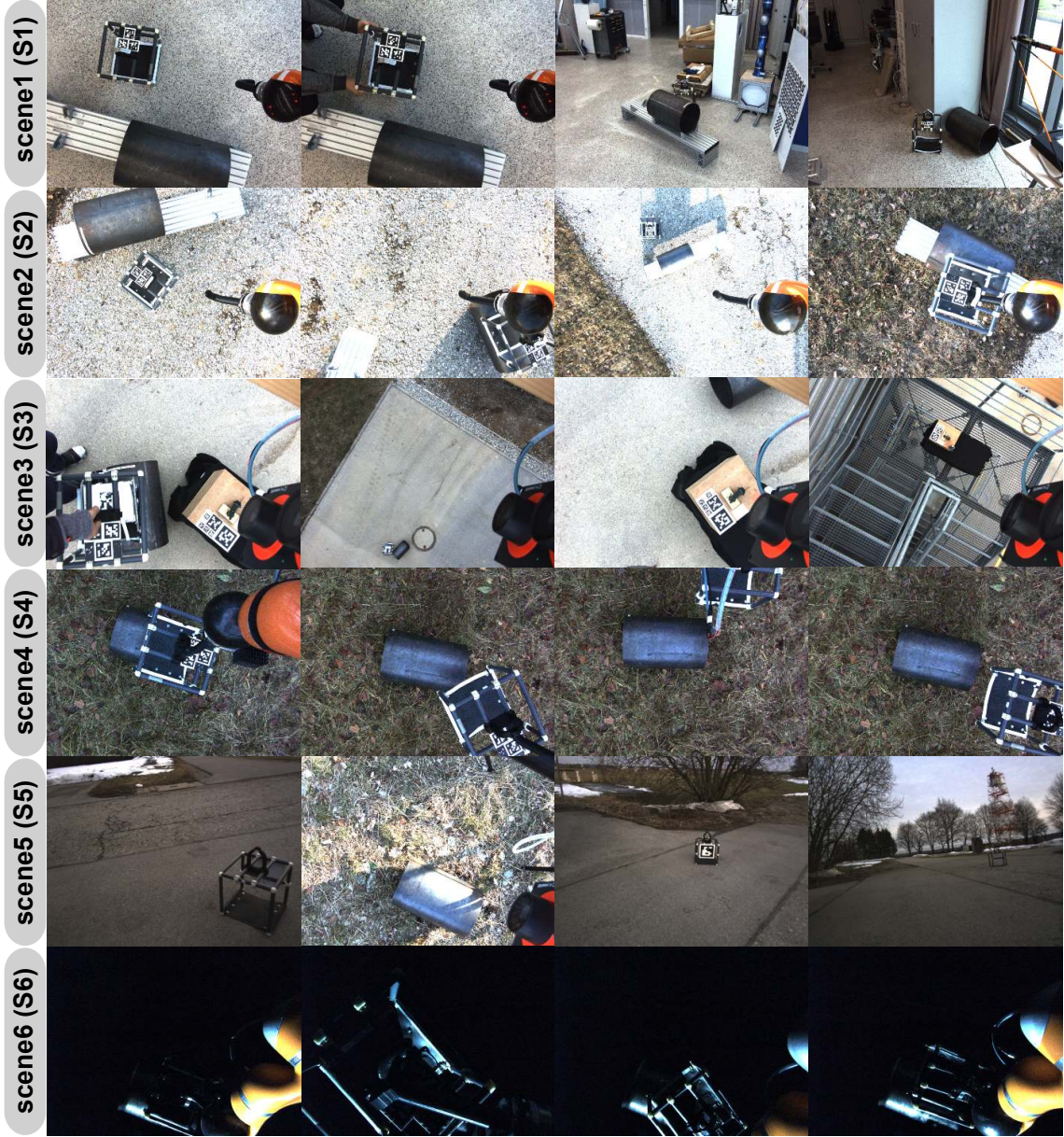
**Figure 8.13.:** Results of active learning (AL) compared to the training set-up that uses 100% of all data over different scenes. AL is used for the acquisition sequence of 5%, 10%, 25%, 50% and 75% of all data. Higher the better for mAP. The results show that one can save upto 75% of data, in order to reach more than 95% of the total performance.

### Bayesian Active Learning for Field Robotics

Lastly, the proposed Bayesian active learning framework is evaluated within the context of field robotics. To recap, the main challenge is to deal with the large variations in the environment, which may hurt the performance of an object detector that has never seen such data in the training set. The natural question to evaluate here is the amount of labeling efforts that the active learning framework can save. As an application of the Bayesian active learning paradigm for field robotics, we focus on the impact of the system performance rather than the algorithmic advances.

**Experiment setup.** To this end, the visual data in various locations and conditions have been gathered. These are not only the (a) laboratory environments, but also (b) the outdoor environments in different locations. These environments are denoted as scenes 1-6 or S1-6, which are visualized in figure 8.8. To evaluate the system performance, the manual annotations within these images have been created. The objects are cage, pipe, and robotic arm. In total of about 20k images, we randomly label 5k images. The splits are performed at the ratio of 7.5:2:0.5 respectively to a pool, test, and validation set. This is to simulate the real-world scenario where the training data is initially limited (e.g. the data collected in summer, and having to test in the winter). We add uniform sampling from pool data (denoted as random) and MC-dropout (Gal & Ghahramani, 2016) as the baseline. While deep ensembles (Lakshminarayanan et al., 2017) are another popular baseline, the suitability to active learning is limited due to the excessive training time. Here, the sampling strategy is chosen from Feng et al. (2022b), and therefore, the only difference between the baseline methods is the uncertainty estimates.

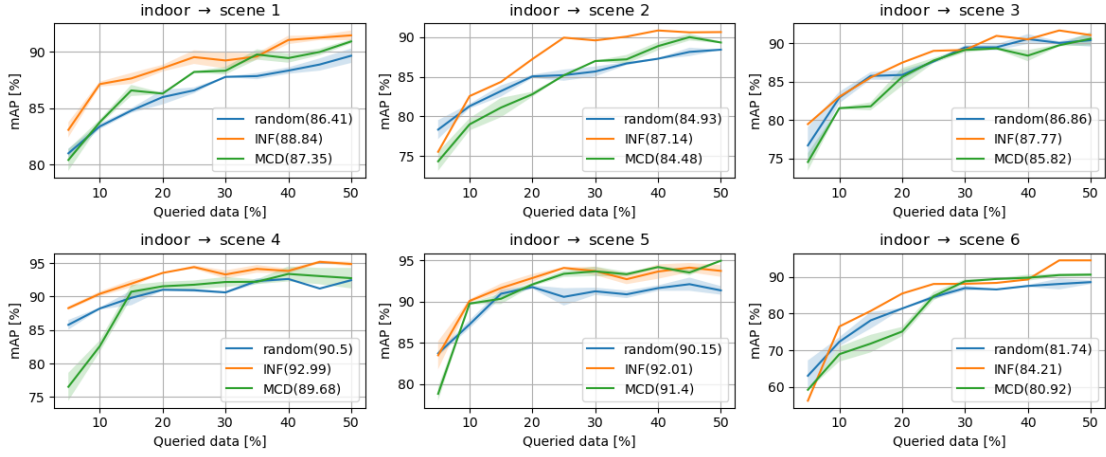
Implementation details are as follows. The Pytorch implementations are used, namely the Retinanet implementation from Detectron2 (Lin et al., 2017) and the official implementation of BayesOD (Harakeh et al., 2020) with slight modifications for better performance. These modifications include the use of Bayesian inference only for the bounding box regression, instead of also applying it to the classification head. The learning rate has also been tuned to obtain better convergence. The Monte Carlo samples of 30 are used for computing the uncertainty estimates, and the rank of 100 and 50 BO iterations are used. The latter is applicable to only the Laplace Approximation, which was applied to all the layers in the Retinanet. On the other hand, only the existing dropout layers within Retinanet have been used for MC-dropout. Such implementations are motivated by the promise of each method. MC-dropout assumes dropout layers and has been popular in practice as one could make



**Figure 8.14.:** Collected sets of pool data from different scenes. Six different scenes cover indoor environments, varying backgrounds and height, and the scenes with snow and at night.

use of existing dropout layers, while Laplace Approximation can directly render every layer as Bayesian, given already trained parameters.

**Results.** Firstly, it is evaluated how much data annotations one can save by comparing the training set-ups that use 100% of the annotated training data against the acquisition sequences of 5%, 10%, 25%, 50%, and 75% of the total data. Repeated sequentially over each scene, the performance of the resulting object detector with mAP as a metric is measured. The test set contains samples from each scene and therefore, this repetition shows how the performance gap due to scene variations is being closed. The results are depicted in figure 8.13. We observe that the gap between the active learner (AL) and the Retinanet with 100% of annotated data (denoted 100%), reduces as we increase the acquisition size



**Figure 8.15.:** Comparisons of the proposed pipeline (INF) with other baselines such as random sampling (random) and MC-dropout (MCD) over six different scenes. The mean mAP during the active learning process is displayed along with the labels of the curve. The standard deviation is shown in shade. Higher the mean mAP, the better.

from 5% to 75% of the total data. In particular, in these scenarios, AL with only 25% of the total annotated data can reach more than 95% of the 100%’s mAP values, which results in saving up to 75% of the annotated data. This result is due to the redundancy in the data. We believe this result can motivate AL for field robotic applications, where the data preparation can be inherently more expensive than the laboratory settings.

Next, the design choices of our active learning pipeline are examined by comparing the method against the selected baselines. The results are depicted in figure 8.15. Here, the transfers between the scenes are assumed. As the robots may operate in different environments, we evaluate by starting the active learning in an indoor scene, and acquiring the data over different outdoor scenes. For all the results, we acquire 5% of the data per iteration, and repeat ten iterations to reach the 50% of all the data. In total, three random seeds are used to compute the standard deviation (also in figure 8.13) for the statistical significance. Examining the mean mAP over all the iterations, the data suggests the performance increase over MC-dropout. The results are consistently observed in several settings with different magnitudes of the improvements. We attribute the reason to the post-hoc nature of our Laplace Approximation based approach. To elaborate, the methods that are based on variational inference, such as MC-dropout, might be at a disadvantage in active learning settings, where each loop involves training a DNN. Naturally, as variational inference rather learns uncertainty during training, finding hyperparameters that deliver good performance over many loops is difficult. On the other hand, post-hoc methods such as ours, the uncertainty estimates are obtained after training the DNN. This decoupling enables us to extensively search for hyperparameters, which is feasible within each active learning loop. Therefore, we interpret these results to show the validity of the design choices of the proposed active learning pipeline. In summary, the key takeaways are the redundancy of the data when training a neural network in dynamic and unstructured environments, and an active learning framework with well-calibrated uncertainty estimates can produce a practical and positive impact by guiding the data preparation steps towards efficiency.





**Figure 8.16.:** Qualitative results for peg-in-hole tasks. Left: approach phase. Middle: precise positioning. Right: successful insertion. Top: overview of the robot's remote workspace, where SAM is suspended by a crane. Middle: the operator view with live video streams, and the created VR, which displays four different view points simultaneously, and provides haptic guidance in both position and orientation. Bottom: close view on the robot's workspace. With the proposed VR-based system, we achieve precise peg-in-hole task with the robot in outdoor environments.

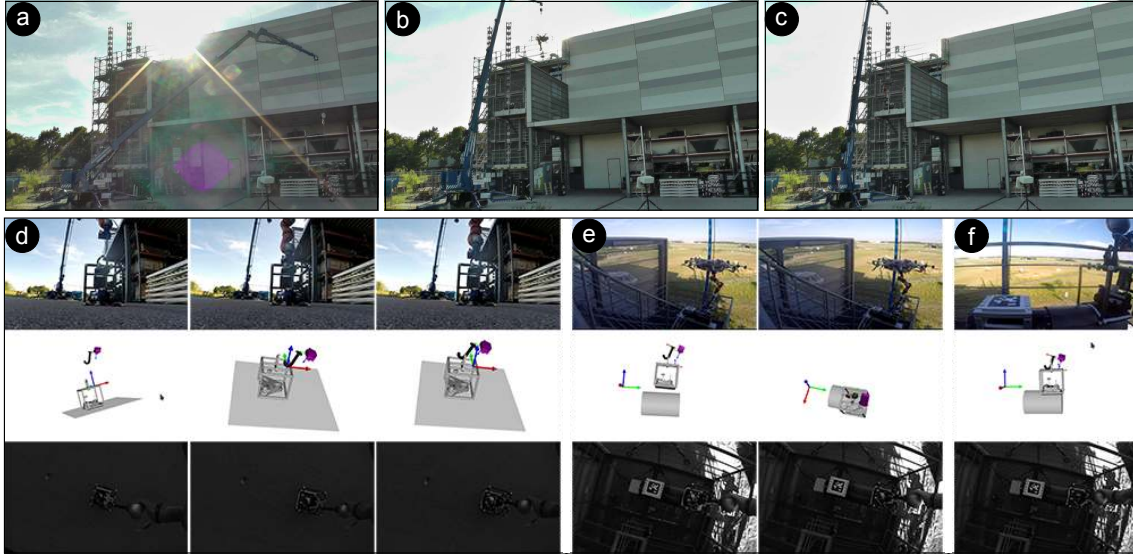
### 8.5.2. Field Testing and User Validation

While the previous focus was on the validation of the methods for VR creation, the flight experiments with SAM are now presented. The main purpose is to evaluate the benefits of the proposed system in advancing aerial manipulation capabilities. To this end, we examine two industrial scenarios in outdoor environments. Then, the robustness of the proposed system is examined by varying environments and users.

**Experiment setup.** The design of our experiment setup is to account for real-world applications of aerial manipulators. As a first step, the description of two industrial scenarios with SAM is provided, which involve the following aerial manipulation tasks:

- **Industrial task 1: peg-in-hole insertion** As one of the benchmarks for manipulation, this task involves inserting an object (attached to the end-effector) into a hole. An example is depicted in figure 8.16. Industrial tasks such as valve opening and closing in high altitude areas, or in-air assembly of structures, require the execution of this task. In this work, the peg-in-hole task with an error margin of less than 2.5mm is considered. This is a challenging task for aerial manipulation, since the robotic arm is on a floating base.
- **Industrial task 2: pick-and-place and force exertion** Two other benchmarks for manipulation are combined, which are pick-and-place and force exertion, in the second task. In particular, our scenario involves deployment and retrieval of an inspection robotic crawler. An example is depicted in figure 8.17. It requires grasping of a cage (as a carrier of the crawler robot), placing the cage on a pipe, and pressing the cage onto the pipe while the crawler moves in and out of it for pipe inspection.

Note that, for the execution of these two tasks, the operator is located far away from the

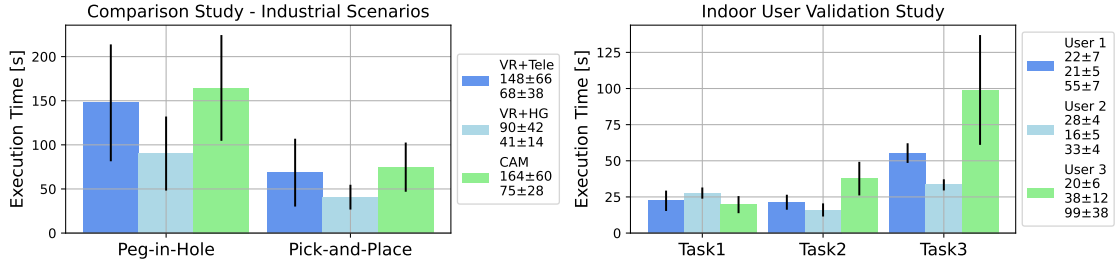


**Figure 8.17.:** Deployment of robotic crawler within an industrial inspection and maintenance scenario. (a-c) The carrier brings the robot from the ground to the remote location. (d) The robot picks the cage that carries the inspection robotic crawler. (e) The robot places the cage on a mock up of an industrial pipe. (f) The robot exerts force on the cage, so that the crawler can roll out of the cage without falling. The operator can use the VR (bottom), which contains 3D information as opposed to 2D camera images (middle). Live video stream is also subject to over and under exposure when under a shadow on a bright day. With the proposed VR-based system, the robot is able to execute advanced aerial manipulation tasks for the considered real world application.

robot without direct visual contact to the workspace of the robot. More concretely, as shown in figures 8.16 and 8.17, the robot operates in an outdoor environment, while its operator remotely commands the robotic arm from a ground station. This simulates a real-world application scenario of teleoperating an aerial robot.

To evaluate the feasibility and benefits of the proposed telepresence system in advancing aerial manipulation capabilities of SAM, four sets of experiments are considered:

- **Set 1: Repetitions of peg-in-hole insertion** Several repetitions of the peg-in-hole insertion task are performed (as shown in figure 8.16). Here, we vary the conditions by executing the manipulation tasks with three modes, namely (a) VR and haptic guidance mode (denoted VR+HG), (b) VR mode (denoted by VR+Tele), and (c) only with live camera streams (denoted CAM). Eight repetitions are performed for each mode, and the total time for a successful execution is chosen as an evaluation metric.
- **Set 2: Repetitions of pick-and-place.** Several repetitions of the pick-and-place task are performed (similar to the environment in figure 8.16 without the crane and the markers on the pipe). Here, we also consider three modes, namely (a) VR and haptic guidance mode (denoted by VR+HG), (b) VR mode (denoted by VR+Tele), and (c) only with live camera streams (denoted CAM). Six repetitions are performed for each mode, with the total time for a successful execution as an evaluation metric.
- **Set 3: User validation.** In a laboratory setting, a user validation study is conducted with three subjects. This is to demonstrate that the considered manipulation tasks can be performed by different users. The considered tasks are force exertion onto a pipe for three seconds (denoted by validation task 1), and also placing a cage on a pipe with and without moving base (denoted as validation tasks 2 and 3 respectively).



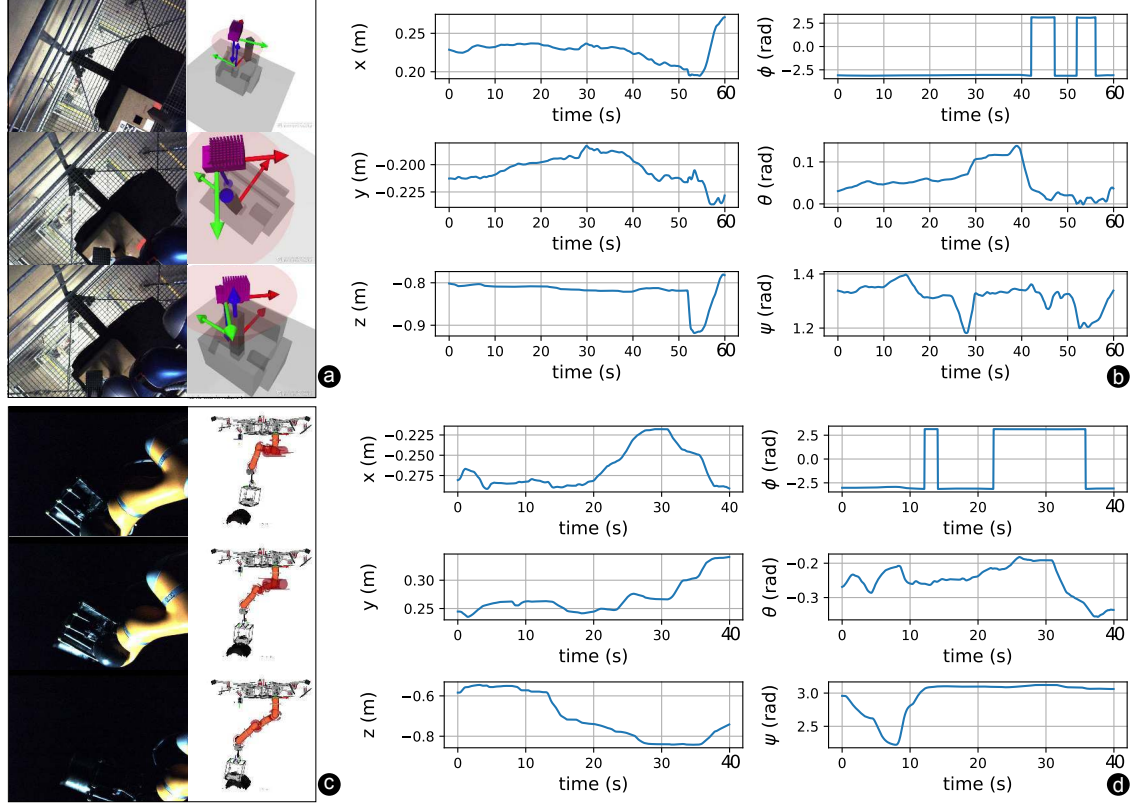
**Figure 8.18.:** Left: the results of the comparison study is depicted, where we compare pure VR-based telepresence (VR+Tele), VR with haptic guidance (VR+HG) and a solution using only a camera (CAM). The statistics are computed over 24 and 16 successful executions in outdoor environments for the peg-in-hole and pick-and-place tasks, respectively. Right: the results of user validation is shown, where three users performed three pre-designed tasks, namely force exertion and placing the cage on the pipe. The statistics are computed over 27 successful executions in an indoor environment. Lower the execution time, better the performance.

With VR and haptic guidance mode, the total time for a successful execution is chosen as our metric.

- **Set 4. Operations at night with VR.** For both industrial tasks, we perform experiments at night. With flash light from an external source, the feasibility and benefits of the proposed system are demonstrated. At night in outdoor environments, this functionality of being able to perform manipulation can increase the range of operation hours, including emergency services for several industrial use cases of aerial manipulators.

With these sets of experiments, we examine the following. For Set 1, the VR+HG mode is examined for achieving manipulation tasks with a high precision. The proposed marker tracking algorithm is utilized here. With Set 2, the benefits of VR in providing depth information to the operator with haptic guidance only for the orientation are examined. The LiDAR-based object pose estimator is utilized here for pipe localization, while the pose of the cage is monitored with the marker tracking method. Set 3 aims for user validation, while with Set 4, we attempt to push the limits of the proposed system. Again, the use case of our system is to augment the live video streams by providing 3D visual feedback and haptic guidance. The use of a VR interface only is not the intended use case of the system. Besides, in industrial scenarios of pipe inspection and maintenance, the pipes are often very long and their inspection points are unknown a-priori. Therefore, the proposed LiDAR-based pose estimation method is used to localize the pipe. This use case justifies the assumption that the object is semantically known, but no geometry is available.

**Results.** The results of Sets 1, 2 and 3 are depicted in figure 8.18. First and foremost, the comparison study of the peg-in-hole insertion tasks with 24 successful executions shows that VR+HG requires the least execution time, while VR+Tele and CAM took similar mean execution times. We note that the executions with CAM used an automated initialization of the end-effector orientation, which was to make the task execution successful. The superiority of VR+HG is expected as the human operator is assisted by the haptic guidance system. Similarly, the comparison study of the pick-and-place task shows a similar trend, where the statistics are computed using 18 successful executions in an outdoor environment. As a third point, from the user validation study, we observed that all the manipulation tasks can be executed by different users. Results from the scenario with a moving base, namely validation task 3, required more time for execution, which indicates that the tasks



**Figure 8.19.:** Qualitative visualization of aerial manipulation performed during the night. (a, c) The camera view and the VR. (b, d) The translation and the orientation of the robots' end-effector. The corresponding pairs are (a,b) and (c,d). These results suggest that the proposed system can extend the operational range of SAM, which further establishes its viability for real world industrial applications. External views are depicted in figure 8.20.

are more challenging with a moving base. Overall, these studies indicate the performance benefits of the proposed system including feasibility and robustness.

The qualitative results of Set 4 are depicted in figure 8.19. The figures display the live view from the eye-in-hand camera, and also from the VR. Lights are provided from an external source and the camera exposures are tuned to achieve a balance between noise, brightness, and stability of streaming. Poses of the end-effector are plotted to illustrate task executions. These plots are also similar for the peg-in-hole and the pick-and-place tasks from previous sets of experiments. Notably in (b) of figure 8.19, peg-in-hole insertion is best characterized in the z-axis between 50s and 60s. In (d) of figure 8.19, the placements are observed in the z-axis between 28s and 35s, while the effects of haptic guidance are shown between 8s and 15s. These experiments show that our system can also work under unfavorable lighting conditions, thereby extending the operational range of the aerial manipulators.

### 8.5.3. Discussion

The results obtained with ablation studies and field experiments suggest successful development and deployment of the proposed VR-based telepresence system for advancing aerial manipulation. For providing both the sense of touch and the sense of vision to the human operator, the proposed system featured not only a haptic device, but also a VR interface



that provides a real-time 3D display of the robot’s workspace as well as haptic guidance. In the experiments, it is shown that the system neither requires any external sensors nor pre-generated maps, copes with the challenges of floating-base manipulation systems, i.e., induced motions of attached sensors due to coupling between the manipulator and the base, fuses multiple sensors whenever appropriate, and has been exhaustively evaluated outside laboratory settings. These features of the proposed VR system are requirements for several industrial applications of aerial manipulation technologies. To the best of our knowledge, using on-board robotic perception only, this work is the first of its kind to demonstrate the feasibility of such a VR-based concept in dynamic and unstructured environments, which includes several outdoor locations, days and nights, as well as different seasons.

To build such a system, several insights are provided, from the identification of practical challenges to their working solutions, both of which are validated using the real data from the robot’s sensors. The pose estimators are subject to a non-holistic view of the objects, which includes loss-of-sight, partial view, and occlusions. For this, we have combined the pose estimators with ego-motion tracking of the environments using real-time SLAM estimates. In the custom datasets that emulate these challenges, the results show that the identified problems can be coped with, which has resulted in the pipelines that meet the requirements of VR creation in accuracy, run-time, and robustness. Moreover, when one aims for a long-term deployment of a learning system in outdoor environments, we find that data collection and preparation become a practical problem. To this end, a pool-based active learning pipeline has been evaluated, which used previous work on uncertainty quantification (Lee et al., 2020b). In field robotics settings, the results show that only 25% of total data is enough to reach 95% of a solution with all data points, and other baseline approaches can be outperformed, overall improving the sampling inefficiency of DNNs.

Overall, the experiments of this work illustrate several benefits of the proposed VR-based telepresence system for advancing aerial manipulation capabilities in real-world applications. Intuitively, a virtual environment allows the human operator to change its sight-of-view, zoom in and out, and provides haptic guidance. In the presented comparison study (with 40 task executions in outdoor environments; a single user), the results show a significant reduction in the total execution time when using the proposed system with haptic guidance. The user validation study (three users with 28 total task executions) suggests that three different users can execute the tasks successfully with varying degrees of performance. Moreover, with the demonstration of the operations at night, the range of operation hours has been extended for the current aerial manipulation systems. All these results are obtained within two industrial scenarios that require advanced aerial manipulation capabilities, namely pick-and-place, force application, and peg-in-hole, which goes beyond a contact-based inspection. Therefore, these results demonstrate the viability of the proposed VR-based telepresence concept for industrial applications in the real world.

## 8.6. Lessons Learned

During the flight campaigns with SAM, we learned a few lessons, which we would like to share with the community. These lessons learned are centered around the proposed VR-based telepresence system. Note that the focus herein is on the use cases of the proposed system, the design choices, and the limitations.

**On use-cases of VR with haptic guidance for aerial manipulation.** The necessity of VR with haptic guidance for SAM (or robots with similar morphology) largely depends on the choice of the haptic device and difficulties of the manipulation tasks. In the initial flight



**Figure 8.20.:** Aerial manipulation at night. The views from an external camera are displayed. Left: SAM performing peg-in-hole insertion task at night. Right: SAM placing a cage onto a metal pipe for the deployment of a robotic crawler. SAM and the objects are highlighted in white.

experiments using only the 2-DoF Space Joystick Rjo in [Lee et al. \(2020a\)](#), the operator could not easily complete the given manipulation tasks when using only live video streams. On the other hand, at the later stages of development, it was much easier for the operator to complete the tasks when we augmented the system with the 6-DOF haptic device Lambda. With the 6-DoF device Lambda and a whole-body controller of the suspended platform to enhance the camera’s field of view, the operators could also complete the tasks using only live video streams, despite slower execution times ([Coelho et al., 2021](#)).

However, while the necessity of VR and haptic guidance may depend on the system and the complexity of tasks, we find that the combination of VR, haptic guidance, and live video stream resulted in the best-performing system. Intuitively, the live video stream can provide situational awareness to the human operator but suffers from over- and underexposure depending on the light conditions, camera jitters due to the movement of the platform under severe winds, lack of complete 3D information, and inability to provide haptic guidance. The proposed VR system can complement the live video stream as it does not degrade with outdoor conditions, provides complete 3D information with an option to change the field of view, and supports haptic guidance. Another benefit is that VR enables seeing the “full model” instead of the limited field of view of the camera at its current position, which includes the configuration of the robotic arm.

**On scene graph versus 3D reconstruction.** The VR creation from robot perception can either rely on scene graph or 3D reconstruction techniques, where the choice of the approach largely depends on the validity of either static-base or floating-base assumptions. For example, ground-based mobile manipulators can first stop, and then perform manipulation. If the scene and objects are static, the relative motion between the sensors and the objects can be easily estimated, and the real-time capability from the perception algorithm is not required. In such a scenario, relying on the outputs of 3D sensors such as RGB-D or stereo would be the simplest option to implement. The robot can map the environments and the objects first to ensure a good field of view, e.g., avoiding occlusions, and then use the map to create a VR. On the other hand, if the relative motion between the sensors and the objects is consistently changing, e.g., in a floating-base system like ours, we find that the scene graph approach can be better suited. The scene graph approach can rely on the object pose estimators that are fast and accurate, and the existing corner cases such as occlusions and loss-of-sight can be handled by using the proposed pipelines. Another consideration is bandwidth, i.e., the object poses require only 6D vectors while streaming point clouds is more expensive. The 6D pose representation can also be plugged in directly

for the shared controllers with position-based visual servoing (as in VR+HG).

**On inherent uncertainty in VR creation.** The proposed VR from robot perception cannot match the reality perfectly. In spite of this limitation, the considered task could be successfully completed even for several challenging outdoor environments. What attributed to the successful deployment of the proposed VR system was identifying when the VR was prone to failures (see figure 8.5). The provided object pose estimators mitigate the identified failure cases by combining a standard object pose estimator with tracking of the environments. Here, the combination is facilitated by a module that identifies the failure cases, e.g., self-evaluation of point cloud processing methods, and missed detection of the markers while using visual-inertial systems. Moreover, in the proposed active learning pipeline, a more explicit representation of uncertainty is used to improve the data preparation steps for our DNN-based component. Therefore, we find that reliability awareness of an algorithm is crucial for robotic systems to achieve complex tasks in dynamic and unstructured environments. This is in line with [Thrun et al. \(2000\)](#).

The current use of DNN’s uncertainty has been off-line, like pool-based active learning, while its use on-board the robot could potentially offer several more benefits. In this regard, combining a real-time uncertainty estimation method ([Lee et al., 2021](#)) with a reliability-aware shared control architecture ([Balachandran et al., 2020](#)), could be an interesting direction of future research for reliable operations of complex systems in unstructured and dynamic environments. Lastly, a full-scale user study is envisioned, which is tailored on telepresence robots with aerial manipulation capabilities in outdoor environments.

## 8.7. Summary

This chapter explores real-world applications of aerial manipulation in dynamic, unstructured environments. We propose a novel telepresence system combining a haptic device for touch and virtual reality (VR) for enhanced visual feedback and haptic guidance. To build this system, we addressed challenges with off-the-shelf methods by developing pose estimation pipelines for industrial objects and a deep active learning pipeline to efficiently gather annotated training data. We validated these methods using robot sensor datasets, examining how each component helps overcome challenges and enables accurate, real-time VR. Methodologically, the key to success was an awareness of the algorithms’ own failures and uncertainty – also known as robotic introspection. One example is the combination of object pose estimation and SLAM, which is facilitated by a module that identified the failure cases. Another example is the active learning pipeline, where information gain is computed from an explicit representation of uncertainty. Using the DLR SAM platform, we conducted over 70 complex aerial manipulation tasks, demonstrating a 1.8x reduction in execution time for pick-and-place and peg-in-hole tasks. The system also functions effectively in low-light conditions, proving its potential for future industrial aerial manipulation applications.



---

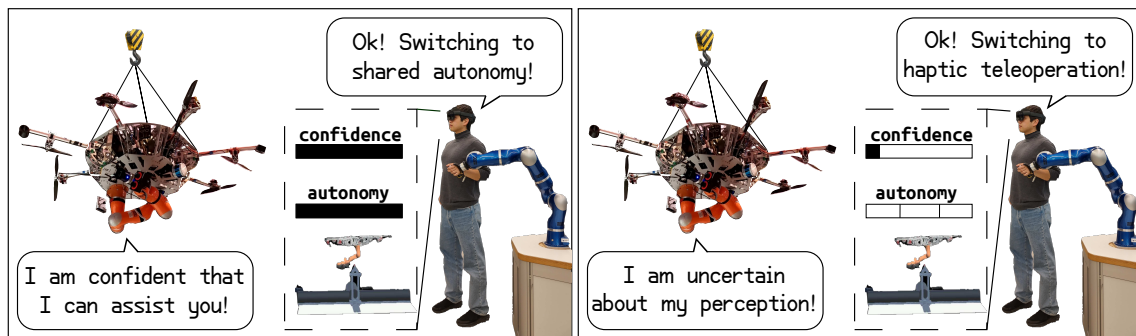
Perceptive Shared Autonomy for Aerial Manipulation under Uncertainty

---

### 9.1. Motivation

In the fall of 1998, an interactive tour guide robot Minerva ([Thrun et al., 2000](#)) was successfully deployed in the Smithsonian Museum. During its two weeks of operation, the robot traversed over 44 km, interacting with thousands of visitors. The key technical enabler was Minerva’s probabilistic algorithms. For the perception of the environments, Minerva relied on a probabilistic algorithm, which localized the robot’s position while estimating its uncertainties ([Fox et al., 2000](#)). All this information was used to generate the robot’s actions such that its mission could be accomplished despite the uncertainties ([Roy et al., 1999](#)). In essence, Minerva showed how probabilistic approaches can enable robust operations in complex environments, where approaches that ignored a robot’s uncertainty could not scale. Inspired by systems such as Minerva, probabilistic approaches in robotics led to sustained progress over the decades ([Dellaert, 2012](#); [Agha et al., 2022](#)).

In this chapter, we propose SPIRIT – a probabilistic system for the teleoperation of aerial manipulators (see figure 9.1). In a similar vein to Minerva, SPIRIT relies on



**Figure 9.1.:** Nowadays, Deep Learning (DL) is being widely used for robotic perception. However, DL cannot always be 100% perfect and can thus fail unexpectedly. A reliable robot needs to cope with such uncertainties in DL. We propose a shared autonomy system that captures uncertainty in DL-based perception and transitions the level of autonomy for robustness.

probability theory for its perception and action. Yet, unlike Minerva, SPIRIT is designed to cope with uncertainty in DL. That is, even if the used DL methods provide wrong predictions when perceiving the environments, SPIRIT’s users can still accomplish the given manipulation tasks. We achieve this by allowing the users to actively transition between semi-autonomous manipulation and haptic teleoperation (“shared autonomy”) (Selvaggio et al., 2021). Concretely, we model the uncertainty of the robot’s DL-based perception. If the uncertainties are low, SPIRIT enables semi-autonomous manipulation for higher performance. When the uncertainties are high, the user can transition to haptic teleoperation, opting for robustness. Here, special emphasis is placed on designing interactive features for the users. Thus, SPIRIT is comprised of a haptic device and a head-mounted extended reality (XR). This human-robot interface intuitively provides the users with haptic and visual feedback about the robot’s current notion of uncertainty.

Originally, SPIRIT was developed for an industrial exhibition. With the first version of the system, we performed live demonstrations of the state-of-the-art aerial manipulation capabilities. Without a single failure, SPIRIT was exhibited to thousands of visitors over five consecutive days. An ample description of this demonstration scenario and the design of SPIRIT is provided in the chapter. In particular, our description focuses on the DL-based perception module because its uncertainty quantification is an open research question (Gawlikowski et al., 2023). For this, we propose a partitioned approach to point cloud registration, where a digital twin of the environment is divided into local regimes. Within these regimes, DL is used to learn and make predictions. To quantify uncertainty, we leverage a Gaussian Process (GP) with the Neural Tangent Kernel (NTK) from chapter 5. Empirically, we first provide ablation studies to motivate the design of SPIRIT’s perception module. Robustness gains and overall system performance are evaluated within a user study with 15 participants and a demonstration of industrial scenarios.

**Contributions and major claims.** Our main contribution is the development of SPIRIT, demonstrating robust aerial manipulation under uncertainty in DL. We coin the term *perceptive shared autonomy* since we modulate the level of autonomy based on uncertainties in DL-based perception (section 9.3). Our user interface with haptic and 3D visual feedback that communicates the robot’s uncertainty is another novelty of this chapter (section 9.3.2). For SPIRIT, we propose a partitioned approach that learns to register point clouds with GP-based uncertainty estimation (section 9.4). This partitioned approach and GP-based uncertainty estimation enable SPIRIT’s development (section 9.5.1). SPIRIT also empirically improves the robustness of the system by achieving the aerial manipulation tasks, even when DL-based perception has unexpectedly failed (section 9.5.2 and 9.5.3). In particular, the operations of the industrial valve establish new state-of-the-art aerial manipulation capabilities in both performance and system reliability.<sup>1</sup>

## 9.2. Related Work

Our contribution is in the area of shared autonomy for aerial manipulation. For this, we build upon probabilistic perception methods with DL. To this end, our work is located within these areas. Table 9.1 summarizes the main novelty of SPIRIT.

**Shared autonomy.** Complete autonomy in aerial manipulation is still a difficult goal and, therefore, we rely on human supervision (Lee et al., 2020a; Coelho et al., 2021; Lee et al., 2024). One classical way to interface a human operator with a robot is through shared autonomy, which combines the benefits of autonomy and direct teleoperation for

---

<sup>1</sup>Link to the project website: <https://sites.google.com/view/robotspirit>.

	Perceptive uncertainty from Deep Learning	Equipped with haptic feedback	Equipped with 3D visual feedback	Validation on floating-base systems
Learning human intent <sup>a</sup>	✗	✗	✗	✗
Adaptive authority allocation <sup>b</sup>	✗	✓	✗	✗
Probabilistic virtual fixtures <sup>c</sup>	✗	✓	✗	✗
The proposed system SPIRIT	✓	✓	✓	✓

**Table 9.1.:** The main novelty of SPIRIT within the state-of-the-art. We bring the idea of (a) using uncertainty in DL-based perception, and (b) interactions between the user and the perception uncertainty with both haptic and XR. These ideas are validated on a moving (floating) base system.

<sup>a</sup>i.e., [Hara et al. \(2023\)](#); [Zhao et al. \(2024\)](#); [Yow et al. \(2023\)](#)

<sup>b</sup>i.e., [Palmieri et al. \(2024\)](#); [Balachandran et al. \(2020\)](#); [Saeidi & Wang \(2018\)](#)

<sup>c</sup>i.e., [Mühlbauer et al. \(2024\)](#); [Aarno et al. \(2005\)](#); [Raiola et al. \(2018\)](#)

semi-autonomous operations ([Selvaggio et al., 2021](#)). One good example is assistive driving, where the onboard autonomy supports the human drivers and reduces their cognitive effort. For shared autonomy, determining the level of autonomy that a human needs is one of the open research questions ([Selvaggio et al., 2021](#)).

To address this challenge, probabilistic methods have been widely investigated. Notable examples are on learning human intent with DL. Recent works ([Hara et al., 2023](#); [Zhao et al., 2024](#); [Yow et al., 2023](#)) use uncertainty estimates in the context of DL-based human intent prediction for improving assistive robot autonomy. Probabilistic methods are also applied for virtual fixtures (VFs) – artificial walls that guide the human operator to achieve the given tasks ([Mühlbauer et al., 2024](#); [Aarno et al., 2005](#); [Raiola et al., 2018](#)). Many investigations on adaptive control strategies with uncertainty quantification ([Palmieri et al., 2024](#); [Balachandran et al., 2020](#); [Saeidi & Wang, 2018](#)) can also be found.

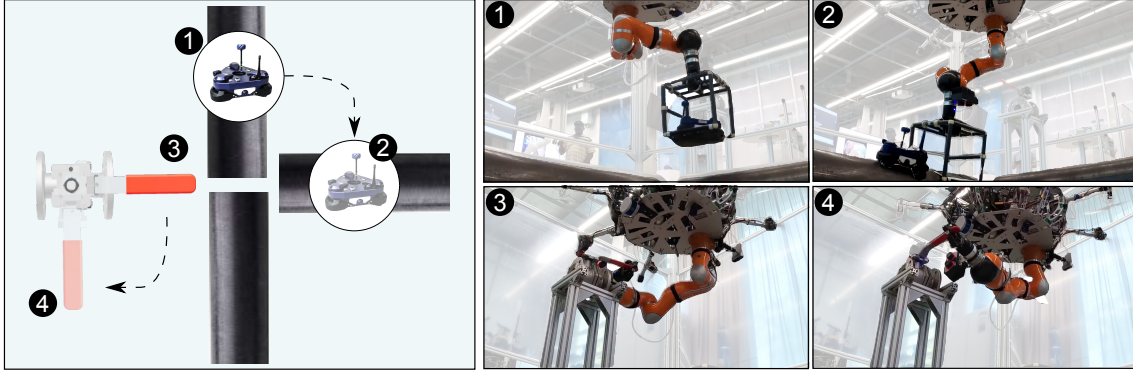
**Aerial manipulation.** Equipping aerial systems with advanced manipulation capabilities is one of the current goals in the field of aerial manipulation ([Ollero et al., 2022](#)). Towards this goal, we examine two advanced aerial manipulation tasks, namely pick-and-place of heavy objects and operations on industrial valves. In particular, to the best of our knowledge, we are the first to demonstrate the forceful operations of industrial flanged ball valves. Unlike valves with circular handles ([Zhao et al., 2023](#); [Nguyen & Alexis, 2021](#)), the challenge is that the valve’s rotation is not around one axis only. We achieve these results while securing robustness against failures in DL-based perception.

**Uncertainty in deep learning.** This work employs and adapts a probabilistic method ([Lee et al., 2021](#)) that provides the so-called sampling-free uncertainty estimates for DL, i.e., the method does not require sampling ([Gal & Ghahramani, 2016](#)) or model ensembles ([Lakshminarayanan et al., 2017](#)), thereby more suited for deployments on a real robot. We refer to the recent surveys for more information ([Gawlikowski et al., 2023](#)). The method is based on GPs with the theory of the Neural Tangent Kernel. Clearly, our goal is not to advance the state-of-the-art in uncertainty quantification. Instead, this work aims to provide a system-level demonstration that the design of an uncertainty-aware perception-action loop can improve robustness against unexpected failures in DL, similar to Minerva.

### 9.3. Perceptive Shared Autonomy under Uncertainty in DL

Among others, we ground the development of SPIRIT in an application within the oil and gas industry. According to Sprint Robotics, more than 825 oil refineries are operating worldwide. The global market was valued at 1,838.46 billion in 2024. These facilities





**Figure 9.2.:** Among many potential applications, we examine two use cases: (a) extending the mobility of an inspection robot via pick and place, and (b) performing manipulation of industrial flange valves. The demonstration of these use cases along with the task sequence (1-4) is illustrated. Left: overview of the tasks. Right: robot performing the tasks from the left image.

require regular inspection and maintenance. For example, only one large refinery has 40,000 km of pipes and 50,000 inspection routines (Ollero et al., 2018). To automate this procedure, robotic crawlers with magnetic wheels are being developed in the industry. However, many pipes are located in difficult-to-reach areas, and mobility for crawler robots is limited. Another problem relates to human safety, especially in the operation of industrial valves. Valves may fail and release hazardous substances or pressure. In 2018, a critical valve malfunction led to an explosion for Husky Energy Inc., which resulted in fatal injuries.

With SPIRIT, we extend the mobility of robotic inspection crawlers and operate industrial valves without human presence in proximity. To show these features, we designed a testbed and a demonstration scenario as depicted in figure 9.2. For the demonstration scenario, first, the robot grasps a cage that hosts an inspection crawler robot. Second, the robot picks up the cage and places the cage onto another pipe. Due to the gap between three pipes in the setup, the crawler robot that uses magnetic wheels cannot traverse from one pipe to another. Thus, the first use case extends the mobility of the crawlers for inspection tasks. Next, the robot attempts to close an industrial valve. For this, the robot grasps the handle of the industrial flange valve, and the robot rotates the valve. Then, once the valve has been closed, the robot releases the grasp while avoiding any collision.

To address these problems in industrial applications, full automation is difficult due to variations in required tasks and environments. Therefore, we rely on teleoperation.

### 9.3.1. The Design of SPIRIT - Shared Autonomy Concept

Yet, for teleoperation of complex systems, the operators have to continuously control the remote robot. Indeed, teleoperation demands high physical and mental effort from human operators (Selvaggio et al., 2021). To resolve such demands from the operators, shared autonomy has been introduced. The key idea is to rely on the autonomy of the robot. When the autonomy tends to fail, the human operator performs corrective measures and actions, thereby reducing the operator's efforts while improving the task completion rate. Among several approaches, this work utilizes the so-called mixed initiative shared autonomy. In this approach, the weights called *authority allocation factor*  $\alpha$  decide how much the task is shared by the human and the robot's autonomy. Mixed initiative can be described by:

$$\dot{s}(t) = f(s(t), a(t)) \quad \text{with} \quad a(t) = \alpha a_h(t) + (1 - \alpha) a_a(t), \quad (9.1)$$

where  $\mathbf{s}$  is the state,  $\mathbf{a}$  is the control input, and the function  $f(\cdot)$  describes the system dynamics. The control input is composed of a weighted addition of human input  $\mathbf{a}_h$  and robot autonomy  $\mathbf{a}_a$ . Within this framework, our key idea is to decide the *authority allocation factor*  $\alpha$  based on uncertainty estimates in DL-based perception.

Concretely, we use two torque-controlled manipulators. One manipulator is attached to a tethered platform (see figure 9.1). Another manipulator is used as a haptic device for teleoperation. The control of the tethered platform and its manipulator is decoupled, i.e., the platform independently stabilizes itself with propellers while the manipulator performs the industrial tasks. The control inputs are Cartesian wrenches  $\mathbf{F}_{\text{wrench}} \in \mathbb{R}^6$  which are converted into joint torques  $\boldsymbol{\tau} = \mathbf{J}^T \mathbf{F}_{\text{wrench}}$ . Here,  $\mathbf{J}$  is the manipulator Jacobian. Now, using the haptic device, the human operator controls the robotic manipulator on the platform. This means, at time  $t$ , the human sends position information  $\mathbf{s}_h$  (velocity analogously). Due to the communication time delay  $T$ , the robot receives  $\bar{\mathbf{v}}_r(t) = \mathbf{v}_h(t - T)$  or equivalently  $\bar{\mathbf{s}}_r(t) = \mathbf{s}_h(t - T)$ . If controller gains are denoted with  $K$ , and  $\mathbf{v}_r, \mathbf{s}_r$  are measured quantities, the commanded wrench  $\mathbf{a}_h(t) = \mathbf{a}_h$  from the operator is:

$$\mathbf{a}_h(t) = \mathbf{a}_h = K_{d,r}(\bar{\mathbf{v}}_r(t) - \mathbf{v}_r(t)) + K_{p,r}(\bar{\mathbf{s}}_r(t) - \mathbf{s}_r(t)). \quad (9.2)$$

At the same time, the robot's onboard autonomy assists the human operator to achieve the desired task. Imagine the visual perception of the robot's target pose  $\mathbf{s}_a$  (or  $\mathbf{v}_a$ ). Then,

$$\mathbf{a}_a(t) = \mathbf{a}_a = K_{d,a}(\mathbf{v}_a(t) - \mathbf{v}_r(t)) + K_{p,a}(\mathbf{s}_a(t) - \mathbf{s}_r(t)), \quad (9.3)$$

is the autonomous action. We use VFs which generate the wrench command  $\mathbf{w}_{\text{wrench}} = \alpha \mathbf{a}_h + (1 - \alpha) \mathbf{a}_a$  to the robot. In return, the robot sends these forces back to the operator.

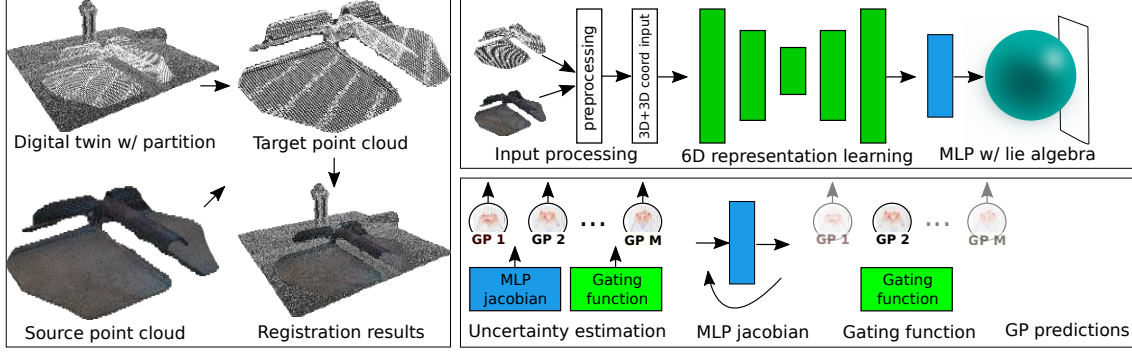
Within this framework, a key concept in SPIRIT is how the authority allocation factor  $\alpha$  is chosen. To recap, VFs are created by the robot's onboard perception system, for which DL is being widely adopted. We assume that the DL-based perception system returns Gaussian distributions with a mean vector and a covariance matrix. Section 9.4 provides an ample description of our perception system. The mean vector is used to create VFs directly. The covariance provides a measure of uncertainty. Using these uncertainty estimates, we decide the level of authority with a metric  $\|\boldsymbol{\Sigma}\|$  and a threshold  $\beta$ :

$$\alpha = 1 \quad \text{if} \quad \|\boldsymbol{\Sigma}\| > \beta \quad \text{else} \quad 0. \quad (9.4)$$

When  $\alpha = 0$ , VFs are turned on for high performance, enabling semi-autonomous manipulation. Otherwise, VF is turned off, opting for robustness through pure teleoperation. Closely following Balachandran et al. (2020), we use the trace as our metric  $\|\boldsymbol{\Sigma}\| = \text{tr}(\boldsymbol{\Sigma})$ , which captures the total variance. The threshold  $\beta$  is chosen from a validation dataset that contains nominal working conditions. In this way, our shared autonomy enables robotic manipulation while addressing uncertainties in DL-based perception.

### 9.3.2. User Interfaces of SPIRIT

In our design of SPIRIT, we equipped the user with both haptic and visual feedback for the interaction with the robot and its uncertainty. As our haptic device, we use a torque-controlled manipulator, which provides an extended workspace. Moreover, we use an XR device, namely Microsoft Hololens2. With Hololens2, we provide both 2D and 3D visual feedback. Live streams from the camera are provided as 2D visual feedback. The robot's state as well as the environment are visualized in 3D. With our setup, the human operator



**Figure 9.3.:** Left: the proposed partitioned approach to point cloud registration. Matching source point cloud to the target point cloud (partitioned from the digital twin) is easier than the use of the entire point cloud from the digital twin of environments. Target point cloud is captured based on robot’s position in the digital twin when performing manipulation tasks. Right: the proposed registration architecture. Lie algebra (a 6D vector) is inferred from neural networks (top). Then, Gaussian Processes (GPs) are utilized for uncertainty estimation (bottom). The underlying GPs are based on the theory of Neural Tangent Kernel. See chapter 5 for more details.

can use hand gestures to zoom in and out, change the sight of view, and further interact with the 3D environment while performing aerial manipulation tasks. We additionally provide telemetry information such as the current level of authority  $\alpha$  and the uncertainty metric  $\|\Sigma\|$ . A foot pedal is additionally integrated to manually change the authority. This ensures that the operator is provided with the full flexibility to operate the remote robot. Figure 9.1 and the video from the project website show the proposed user interface.

## 9.4. Uncertainty-Aware Perception in SPIRIT

Perception module of SPIRIT has two objectives. First, the module has to provide uncertainty estimation for shared autonomy. Second, the robot’s states and models of the environments are used in our 3D visualization software with XR. We assume that the robot is equipped with a 3D sensor to obtain the point clouds  $\mathbf{P} = [\mathbf{p}_1, \dots, \mathbf{p}_P] \in \mathbb{R}^{3 \times P}$ . The 3D sensor is installed without occlusions to the environments. In the given industrial scenario, we can assume known objects such as a cage and a pipe inspection robot. For visual perception of these objects, we utilize fiducial markers of Wang & Olson (2016) due to their accuracy and robustness. There also exist objects that belong to the environments, such as industrial pipes and valves. For these objects, we cannot assume the availability of fiducial markers since industrial sites are not equipped with markers for robots.

### 9.4.1. Learning to Register Point Clouds with Digital Twins

On the other hand, digital twins of industrial sites are becoming a reality (Xiangdong et al., 2020). Like high-definition maps on roads for autonomous driving, 3D models of industrial sites with pipelines and industrial valves can now be used. A key advantage is the simplification of the robot’s perception system. Related works had to rely on sophisticated pipelines with odometry, object detection, 3D reconstruction, and more (Lee et al., 2020a, 2024). Instead, under the aforementioned assumptions, the core functionality simplifies to a point cloud registration solver that aligns two point clouds. Here, the first point cloud is the current sensor measurements, which are then aligned to a reference point cloud from

the digital twin. The resulting pose for alignment localizes the robot in its environment. Because objects like pipelines and valves are static within the environments, localization of the robot also enables 6D object pose estimation through a digital twin.

However, point cloud registration in complex environments can be challenging. For example, finding the points from one point cloud that correspond to points in another point cloud, often referred to as the correspondence estimation step in point cloud registration, might be difficult if two point clouds significantly differ. The digital twin contains a 3D model of the entire environments, while the robot’s sensors may only provide local measurements. As these local measurements constitute only a small portion of the entire environments, finding correspondences may not be trivial. To this end, we propose a partitioning approach. Our key idea is to partition the 3D model of the entire environments, so that the given point cloud registration problem is further simplified. That is,

$$\mathbf{T}^* = \operatorname{argmin}_{\mathbf{T}} \rho(\mathbf{T}(\mathbf{Q}), \mathbf{P}), \quad (9.5)$$

where  $\mathbf{Q} = [\mathbf{q}_1, \dots, \mathbf{q}_Q] \in \mathbb{R}^{3 \times Q}$  is a partitioned point cloud from the digital twin. Here, when manipulating, tethered robots attempt to regulate around one position. The point cloud is therefore obtained from the digital twin, considering the robot’s position for manipulability (see figure 9.3).  $\mathbf{T}$  is a transformation and  $\rho$  is a distance measure.

A digital twin also enables the collection of training data in order to address the registration problem with DL. For this, photorealistic synthesizers such as BlenderProc (Denninger et al., 2023) provide a plugin to generate point clouds that emulate sensor measurements. Within this context, we propose our DL-based pipeline with the following steps. Firstly, for two point clouds  $\mathbf{P}$  and  $\mathbf{Q}$ , we obtain features  $\mathcal{F}_{\mathbf{P}} = (\mathbf{f}_{\mathbf{P}_1}, \dots, \mathbf{f}_{\mathbf{P}_P})$  and  $\mathcal{F}_{\mathbf{Q}} = (\mathbf{f}_{\mathbf{Q}_1}, \dots, \mathbf{f}_{\mathbf{Q}_Q})$ . Then, we use the nearest neighbor to generate the correspondences  $\mathcal{M} = \left\{ (i, \operatorname{argmin}_j \|\mathbf{f}_{\mathbf{P}_i} - \mathbf{f}_{\mathbf{Q}_j}\|) \right\}$  for  $i = 1, \dots, P$  given  $j = 1, \dots, Q$ . We utilize these correspondences of features as a sparse input  $\mathbf{x}$  to a neural network  $f_{\theta}$ :

$$\mathbf{y} = f_{\theta}(\mathbf{x}) \text{ with } \mathbf{x} = \begin{bmatrix} \mathbf{f}_{\mathbf{P}_1} & \dots & \mathbf{f}_{\mathbf{P}_{|\mathcal{M}|}} \\ \mathbf{f}_{\mathbf{Q}_{j_1}} & \dots & \mathbf{f}_{\mathbf{Q}_{j_{|\mathcal{M}|}}} \end{bmatrix}. \quad (9.6)$$

Here, the indices  $J$  define the correspondences  $\mathbf{f}_{\mathbf{P}_i} \leftrightarrow \mathbf{f}_{\mathbf{Q}_{j_i}}$ . We utilize a U-net like 6D convolutional network with skip connections across the network (Choy et al., 2020). A multi-layer perceptron is added later to obtain a 6D representation called lie algebra  $\mathbf{y}$  that can be mapped into transformation matrices (Barfoot, 2024). Lie algebra is in Euclidean space as opposed to transformation matrices or Lie groups, allowing us to use Gaussian distributions. The architecture is depicted in figure 9.3. After these steps, we can optionally use the iterative closest point (ICP) algorithm to refine the obtained results further. The robot’s onboard visual-inertial SLAM can also be used to track the environments and provide estimates of the target pose at a faster rate (Lee et al., 2020a). The model is trained with a mean squared error on lie algebra. The training data is obtained with the digital twin using the proposed partitioning idea for a zero-shot sim2real transfer.

#### 9.4.2. Uncertainty Estimation with Gaussian Processes

Next, we describe how we estimate uncertainties. Typically, epistemic and aleatoric uncertainty are commonly discussed types of uncertainty (Gawlikowski et al., 2023). Epistemic uncertainty captures incomplete information that a model encapsulates, while aleatoric

uncertainty refers to the inherent randomness in a learning process. Imagine noise in the annotations of training data for supervised learning. We model both types of uncertainties.

To do so, we utilize Gaussian Processes (GPs), which are often used for decision-making under uncertainty (Rasmussen, 2003). GPs capture both types of uncertainties. Concisely, let us consider a data tuple, consisting of inputs  $\mathbf{x}$ , noisy outputs  $\mathbf{y}$ , and a function that relates them:  $\mathbf{y} = h(\mathbf{x}) + \epsilon$ . The noise is assumed to be isotropic:  $\mathcal{N}(0, \sigma_n)$ .  $\sigma_n$  is aleatoric uncertainty. With GPs, the function values at these points are a Gaussian with mean  $\mathbf{y}$  and covariance  $\mathbf{\Sigma}$ . The covariance is specified by the so-called kernel function  $k(\cdot, \cdot)$ . Now, given a new input  $\mathbf{x}^*$ , GPs make predictions, resulting in  $\mathcal{N}(\mathbf{y}^*, \mathbf{\Sigma}^*)$  by:

$$\begin{aligned} \mathbf{y}^* &= \mathbf{k}_*^T (\mathbf{K} + \sigma_n \mathbf{I})^{-1} \mathbf{Y}, \\ \mathbf{\Sigma}^* &= \mathbf{k}_{**} - \mathbf{k}_*^T (\mathbf{K} + \sigma_n \mathbf{I})^{-1} \mathbf{k}_* + \sigma_n \mathbf{I}, \end{aligned} \quad (9.7)$$

Here,  $\mathbf{Y}$  is the column vector of noisy outputs  $\mathbf{y}$  at training,  $\mathbf{k}_* = k(\mathbf{x}^*, \mathbf{X})$  is the row vector of covariances between new input  $\mathbf{x}^*$  and training data  $\mathbf{X}$  (column vector), and  $\mathbf{K} = k(\mathbf{X}, \mathbf{X})$  is the covariance matrix from the training data. Likewise,  $\mathbf{k}_{**} = k(\mathbf{x}^*, \mathbf{x}^*)$ . The kernel function defines the covariances, capturing similarity between pairs of points in input space. Intuitively, GPs capture uncertainty estimates by comparing the similarity between training data and new test input through kernel functions.

To utilize GPs for quantifying uncertainty in DL, we employ Lee et al. (2021). Therein, the uncertainty estimates of neural networks are estimated using a variant of GPs called local GPs. This method relies on a specialized kernel function called the Neural Tangent Kernel (NTK), which enables the combination of GPs and DL in a post-hoc manner. That is, a neural network can make predictions  $\mathbf{y}^* = f_\theta(\mathbf{x}^*)$ . Its uncertainty is estimated using GPs. To explain, we define a pseudo output of neural networks:  $\tilde{\mathbf{y}} = \mathbf{J}_f(\mathbf{x})\hat{\boldsymbol{\theta}} + \eta$ , where  $\mathbf{J}_f$  is the first derivative of output  $\mathbf{y}$  w.r.t the parameters of neural networks  $\hat{\boldsymbol{\theta}}$ .  $\eta$  is a loss-dependent residual term. Due to the linear mapping, the covariance of  $\tilde{\mathbf{y}}$  is equal to that of  $\mathbf{y}$ . Thus, we define a local GP on  $\tilde{\mathbf{y}}$  to obtain the uncertainty estimates. Note that, in theory (Lee et al., 2021), a GP cannot be defined on the output of neural networks  $\mathbf{y}$  directly. Now, local GPs consist of  $M$  GP models and a gating function  $g(\mathbf{x})$ . The gating function decides which GP model to pick for both training and predictions. That is,

$$\tilde{\mathbf{y}} = \sum_{m=1}^M g_m(\mathbf{x}) h_m(\mathbf{x}) + \epsilon_m \text{ and } k(\mathbf{x}, \mathbf{x}) = \mathbf{J}_f(\mathbf{x})^T \mathbf{J}_f(\mathbf{x}) \quad (9.8)$$

We assume a hard partitioning, i.e.,  $g(\mathbf{x}) = 1$  in one coordinate for each input. The NTK is also defined, which is the outer product of the neural network Jacobian. More details can be found in chapter 5. We define our gating function according to the partitioning of the digital twin. Marginal log likelihood is optimized to train GPs. If ICP is used for refinement,  $\max(\cdot)$  function can be used to compute the final  $\|\mathbf{\Sigma}\| = \text{tr}(\mathbf{\Sigma})$ .

Local GPs are well suited for the considered applications. First, local GPs are equipped with the ability to provide sampling-free uncertainty estimates for run-time efficiency. This means our method does not introduce significant run-time overhead due to uncertainty quantification. Second, local GPs offer the ability to capture non-stationary aleatoric uncertainty estimates. For each partitioned regime of the digital twin, we can infer a different aleatoric uncertainty term. Lastly, local GPs contain one GP per partitioned regime, obtaining scalability gains due to the division of the input datum. Figure 9.3 depicts the prediction mechanisms with local GPs, which involve mainly three steps.



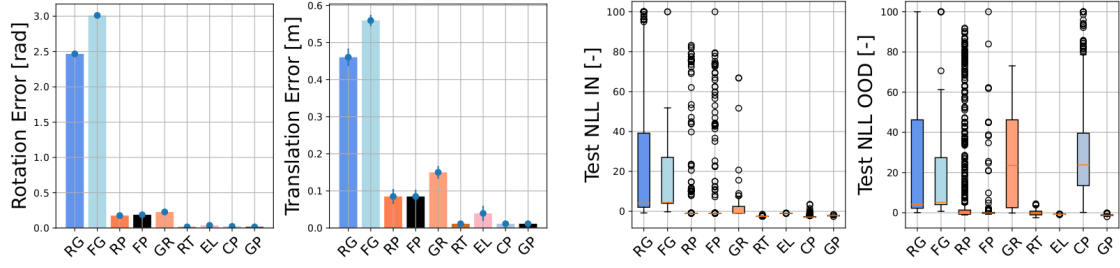


Figure 9.4.: Results of ablation studies. Lower the better.

## 9.5. Experiments and Evaluations

### 9.5.1. Ablation Studies on SPIRIT’s Perception

First, we provide ablation studies and comparative assessments on SPIRIT’s perception module. Specifically, we analyze our design choices in the proposed approach to partition the point cloud registration problem with uncertainty estimates. For this ablation study, we plug in our digital twin to Blenderproc software in order to generate photo-realistic images with point clouds. An advantage of using such synthetic data is the availability of ground truth for noise-free evaluation. Furthermore, the data can also be used to train DL methods, where depth-only approaches suffer less from the sim-to-real gap; e.g., we adapt zero-shot transfer in all our real-world experiments. Emulating our demonstration scenario in figure 9.2, we pick four regimes of interest (or partitions equivalently). For each regime, we sample 1200 viewpoints from a sphere with a variance of 0.2 m, from which we split the train and test set with a ratio of 0.8. Regarding the evaluation metrics, we choose run-time, mean squared error (MSE) over rotations and translations, and test negative log likelihood (NLL) for the measure of uncertainty (Gawlikowski et al., 2023).

In total, we chose eight baselines for ablation studies. First, we implement RANSAC-based global registration (RG) and fast global registration (FG). Then, next two baselines are their combinations with the proposed partitioned approach with both RANSAC (RP) and fast global registration (FP). In this way, we can examine the influence of our partitioned approach. Then, while utilizing the partitioned approach, we compare deep global registration with the weighted Procrustes (GR Choy et al. (2020)) and the proposed architecture (RT). This comparison can motivate the design of our underlying DL architecture. For these deterministic methods, we modeled homoscedastic aleatoric uncertainties. Finally, with RT, we ablate different uncertainty estimation methods. We note that the goal is to examine the performance in our specific application scenario. Therefore, we compare the proposed approach with NTK-based GPs (GP) against two recent baselines, namely evidential learning (EL) (Amini et al., 2020) and conformal predictions (CP) (Fontana et al., 2023). Unlike MC-dropout and Deep Ensembles (Gawlikowski et al., 2023), these baselines are sampling-free methods and have gained popularity for robotics in recent years because of run-time efficiency (Amini et al., 2020; Fontana et al., 2023).

The results are depicted in figure 9.4. Firstly, improvements in accuracy are observed between non-partitioned and partitioned baselines. Moreover, the learning baselines yielded

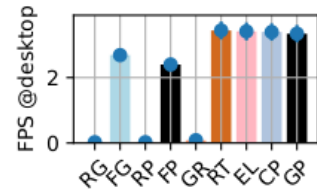


Figure 9.5.: Run-time analysis.

Set 1	Vanila-teleop	SPIRITv1	SPIRITv2
Success rate [%] $\uparrow$	86.66 $\pm$ 0.000	93.33 $\pm$ 0.000	<b>100.0<math>\pm</math>0.000</b>
Time [s] $\downarrow$	160.46 $\pm$ 144.1	149.1 $\pm$ 112.3	<b>61.86<math>\pm</math>46.03</b>
Forces [N] $\downarrow$	11.66 $\pm$ 1.325	<b>10.89<math>\pm</math>0.867</b>	11.72 $\pm$ 1.414
Torques [Nm] $\downarrow$	6.674 $\pm$ 0.748	<b>6.351<math>\pm</math>0.381</b>	7.442 $\pm$ 0.810
NASA TLX [-] $\downarrow$	11.58 $\pm$ 3.715	11.71 $\pm$ 2.118	<b>7.422<math>\pm</math>2.613</b>
SUS score [-] $\uparrow$	51.00 $\pm$ 25.73	52.66 $\pm$ 19.41	<b>71.33<math>\pm</math>16.77</b>
Set 2	Vanila-VF	SPIRIT	p (set 1 / 2)
Success rate [%] $\uparrow$	40.00 $\pm$ 0.000	<b>100.0<math>\pm</math>0.000</b>	(N/A / N/A)
Time [s] $\downarrow$	335.0 $\pm$ 202.2	<b>78.33<math>\pm</math>72.77</b>	(0.0189/0.0249)
Forces [N] $\downarrow$	13.70 $\pm$ 1.018	<b>10.86<math>\pm</math>1.586</b>	(N.S/0.0010)
Torques [Nm] $\downarrow$	7.202 $\pm$ 0.449	<b>6.679<math>\pm</math>0.903</b>	(0.0004/N.S)
NASA TLX [-] $\downarrow$	14.18 $\pm$ 3.416	<b>8.288<math>\pm</math>4.137</b>	(0.0002/0.0003)
SUS score [-] $\uparrow$	35.83 $\pm$ 20.92	<b>65.50<math>\pm</math>24.10</b>	(0.0245/0.0016)

**Table 9.2.:** Results from the user study. Set 1 evaluates the performance of shared autonomy while set 2 is designed to evaluate the system reliability against the failures of DL-based perception.

smaller errors. Both observations motivate the proposed utilization of a digital twin. Secondly, regarding the uncertainty estimates, better test NLL is noted for probabilistic baselines. More impact is on the failure cases (OOD) rather than nominal conditions (ID). In general, EL resulted in degradation of accuracy, which could be due to its non-standard loss function. In our experiments, conformal prediction also did not result in better uncertainty estimates. GP was not the most efficient method in terms of run-time. Yet, the method resulted in better test NLL and accuracy with similar run-time. These results motivate the design choices on SPIRIT’s perception, and only GP could maintain similar NLL at OOD.

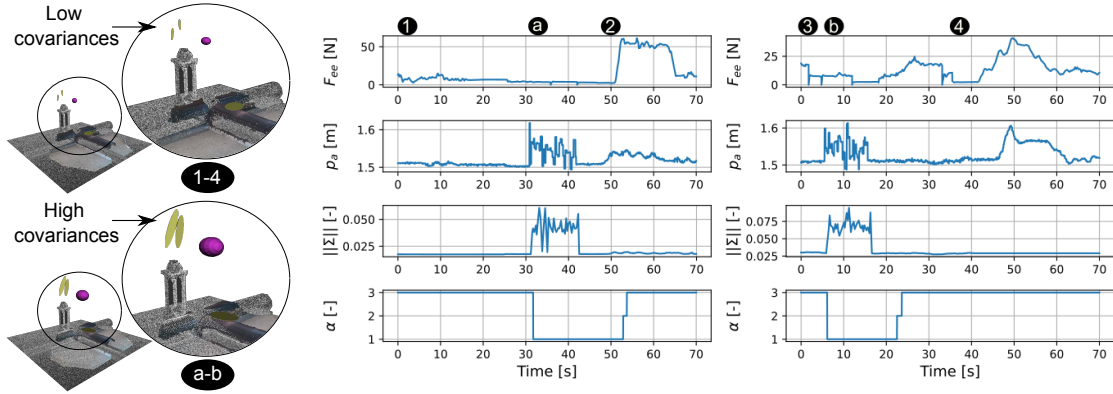
### 9.5.2. User Studies on SPIRIT’s Shared Autonomy

Next, we conduct a user study in order to evaluate SPIRIT in assisting a human operator for the given manipulation tasks, while coping with uncertainty in DL. A testing sample of 15 consisting of 12 male and three female is considered. On average, the subjects had an age of 30.93 years (SD=3.76 with a range 26-39). All the subjects had little to no experience in teleoperation and XR. An informed consent form was signed for the study. We instructed the subjects to complete the valve grasping task (see figure 9.2) remotely. That is, no direct view of the robot was provided while executing the tasks.

We divided the study into two sets. First, we examine three modes of operation, namely bilateral haptic teleoperation with XR (SPIRITv1), unilateral teleoperation without XR (vanila-teleop) and the proposed system (SPIRITv2). This comparison is to evaluate the benefits of shared autonomy. The mode vanila-teleop is included to evaluate our user interface, where the subject was only provided with a video stream on a computer monitor. The order was randomized. Second, we evaluate the robustness against the failures in the DL-based perception. Two modes are examined, namely the proposed system and a system without uncertainty (vanila-VF). Here, we manually fail the perception by perturbing a point cloud with a noise (0.25 $\pm$ 0.5). While SPIRIT can change its authority, vanila-VF would result in a complete failure. In order to avoid breaking the hardware, we emulated the failure by introducing a sinusoidal disturbance to the VF with a 10 Hz frequency and 0.1 m amplitude. The order was again counterbalanced across the subjects.

Firstly, the concept of SPIRIT, its core features, and the tasks to be completed were briefly introduced to the subjects. Then, the subjects were asked to complete a demographic questionnaire. Initially, the subjects were introduced to the system and trained to move the





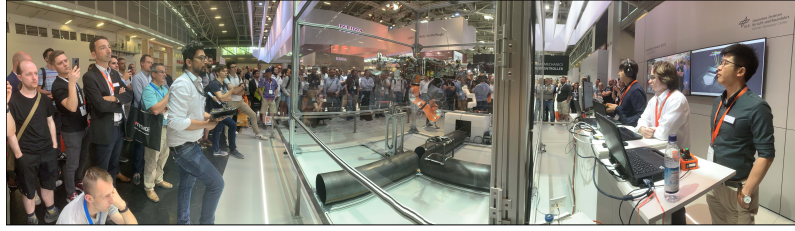
**Figure 9.6.:** Results from the demonstration of industrial scenarios. Left: the point cloud registration is visualized with covariances. Blue ellipse represent the covariance over the translations while yellow ellipse depict the covariance over rotations for each axis. Middle: pick-and-place of heavy objects; (1) grasping the object, (a) failures in DL-based perception over 10 s, and (2) pick-and-place with over 50 N end-effector forces. Right: forceful operation of flange valve; (3) grasping of the valve bar, (b) failures in DL-based perception while grasping, and (4) closing of the valve with about 50N end-effector forces (norm). Example video is on our website.

robot in free space until they felt ready. After the training, the aforementioned experiments began. For each set of experiments, subjects were administered the NASA Task Load Index (TLX) and System Usability Scale (SUS), which qualitatively measure the overall workload and usability. At the end of the study, a final questionnaire with several items on situational awareness was used. The questionnaire ranged from ‘Does not apply at all’ to ‘Completely applies’ with an eight-point scale. Completion time, success rate, mean forces, and torques were measured as quantitative measures. A task was assumed to have failed after 500 s or if the decision was made to stop. Analysis of variance (ANOVA) and mean with standard deviation were calculated for testing the statistical significance.

The results are presented in table 9.2. From set 1, we examine the benefits of shared autonomy (SPIRITv2) in terms of success rate, completion time, workload (NASA TLX) and usability (SUS score). Moreover, when compared to vanilla-teleop, the benefits of haptic and elaborated visual feedback (SPIRITv1) are observed in quantitative metrics such as success rate and completion time. Mean forces did not pass the statistical significance testing (N.S). The subjects used the least torque in SPIRITv1 and the most torque in SPIRITv2. Mean forces and torques may measure the physical workload. At the same time, in conjunction with completion time, higher forces and torques may indicate the confidence of the users with the system. Overall, the results from set 1 provide evidence that our shared autonomy can improve the success rate and completion time of the users. The subjects also rated the shared autonomy as more usable with less workload.

Furthermore, we summarize the results of set 2 in table 9.2. We observe that SPIRIT achieves superior results in all the measures. In particular, the users of SPIRIT accomplished the given tasks with a 100% success rate, despite the failures in the DL-based perception system. We note that the success rate of 40% with vanilla-VF may not be feasible in real failure cases of DL-based perception. Imagine the jump in the estimates of the target’s rotation, which can damage the robot’s hardware or result in a collision with the environment. On the other hand, SPIRIT estimates the uncertainties in the DL-based perception. In case of failures, the authority is shifted to pure teleoperation. Then, the user can still accomplish the tasks without the support of the autonomous system. Even

**Figure 9.7:** Panoramic view of the exhibitions at Automatica 2023 – one of the world’s largest industrial fair on robotics and automation. Awarded the finalist of KUKA Innovation Award 2023.



with different users, these results indicate that SPIRIT can improve the reliability of the overall system that relies on DL for their perception.

In contrast, we also found certain limitations from the user study. In the situational awareness questionnaires, the subjects mainly rated to be aware of the robot’s positions and actions ( $5.062 \pm 1.611$ ), capable of predicting the robot’s action ( $5.187 \pm 1.515$ ) as well as the degree of control they had on the robot ( $6.125 \pm 1.204$ ). Most importantly, the subjects generally felt that both haptic ( $6.312 \pm 0.946$ ) and XR interface ( $6.750 \pm 1.064$ ) helped them in increasing the awareness of the robot’s confidence and authority allocations, which validates the design of our user interface. However, two subjects noted in their feedback that (a) excessive information was present in the user interface and (b) they would have preferred stronger haptic feedback. Thus, in the future, we envision the design of a more flexible user interface which can adapt according to the preferences of humans individually.

### 9.5.3. Completion of Industrial Scenarios with SPIRIT

Finally, we showcase SPIRIT for applications of the inspection and maintenance in oil and gas refineries (previously detailed in section 9.3; see also figure 9.7). The scenario involves the operations of flange valves and pick-and-place of heavy objects. We again emulate perception failures by introducing the same noise to the incoming point clouds while the human operator is performing manipulation tasks in semi-autonomous mode with VFs. The results are depicted in figure 9.6. For both tasks, the prediction quality ( $p_a$ ) decreases significantly when the failure is introduced. Despite this failure, the results indicate the accomplishment of the given manipulation tasks. The given uncertainty criterion ( $\|\Sigma\|$ ) is able to capture the failure, and thus the level of autonomy ( $\alpha$ ) is modulated accordingly. We further note that the robot exerted about  $50N$  of wrench forces for both tasks, which advances the state-of-the-art performance in forceful aerial manipulation. These results suggest that SPIRIT is able to perform aerial manipulation under uncertainty in DL. Within an application scenario, SPIRIT advances the state-of-the-art aerial manipulation in terms of both performance and the system’s reliability against perception failures. In this way, in a similar vein to Minerva but within a modern context, SPIRIT shows the relevance of probabilistic approaches when employing DL methods in complex real-world settings.

## 9.6. Summary

In this work, we present SPIRIT – a perceptive shared autonomy system for robust aerial manipulation under uncertainty in deep learning (DL). By transitioning the level of autonomy based on uncertainty estimates, we demonstrate how DL can be more safely integrated into robotic systems. From comparative assessments, user study with 15 participants, and demonstrations of industrial scenarios, ample evidence illustrates that SPIRIT enables robotic manipulation under uncertainty in DL, enhancing the system’s reliability against

unexpected failures in DL-based perception. In the near future, we hope to take SPIRIT for industrial pilot studies and advance its functionalities towards real industrial applications. In particular, an interesting future venue is analyzing real-world failure cases of DL-based perception in several field experiments and extending SPIRIT for long-term deployments.



# Part IV.

## Epilogue

“No problem can be solved from the same level of consciousness that created it” – Albert Einstein.



This chapter presents the applications of probabilistic methods and systems in the DLR’s ongoing projects. Concretely, we extend both the methods and systems from the previous chapters and show how these works are being used in the DLR’s research and development activities. Three application domains are considered. First, we examine the applications to future manufacturing (“Industry 4.0”). In the so-called fourth industrial revolution, collaborative robots are enabling safe interactions with humans for more flexible and versatile production. Second, we examine robotic planetary exploration. DLR, as a space agency, is focused on demonstrations and spaceflight missions. This chapter presents results from the space analog demonstration mission called ARCHES. Lastly, we examine the applications to terrestrial assistance, where intelligent service robots in everyone’s homes are imagined. In particular, results from activities on the adult-size humanoid robot, TORO, are presented. Involvement in many of these projects resulted in experiences quantifying uncertainties for real-world applications. We impart the knowledge in the form of a probabilistic programming method called Paula. Details behind Paula are discussed.

## 10.1. Yantra: A Robot Apprentice for Textile Manufacturing

In the context of future manufacturing, we present a system called Yantra<sup>1</sup>. The key idea behind Yantra is a collaborative robot that supports traditional textile manufacturing. In countries like India, Japan, and Indonesia, traditional textile manufacturing is characterized by craftsmanship within a micro-factory setup. To manufacture traditional textiles, craftsmen utilize a variety of manufacturing processes, often involving a combination of traditional techniques and modern tools. One of the most tedious processes is called ikat, which is a preparatory phase in traditional textile manufacturing. Ikat involves the processes of (a) yarn preparation where threads are marked and painted, and (b) weaving which refers to the process of loading the painted threads into a hand-loom. While ikat is a textile art that results in uniquely patterned fabrics, the processes rely purely on craftsmen. With Yantra, we attempt to develop a collaborative robot that can work alongside the craftsmen for ikat. In this way, the robot performs the repetitive tasks while the craftsmen focus on

---

<sup>1</sup>Yantra won the finalist of KUKA Innovation Award 2024.



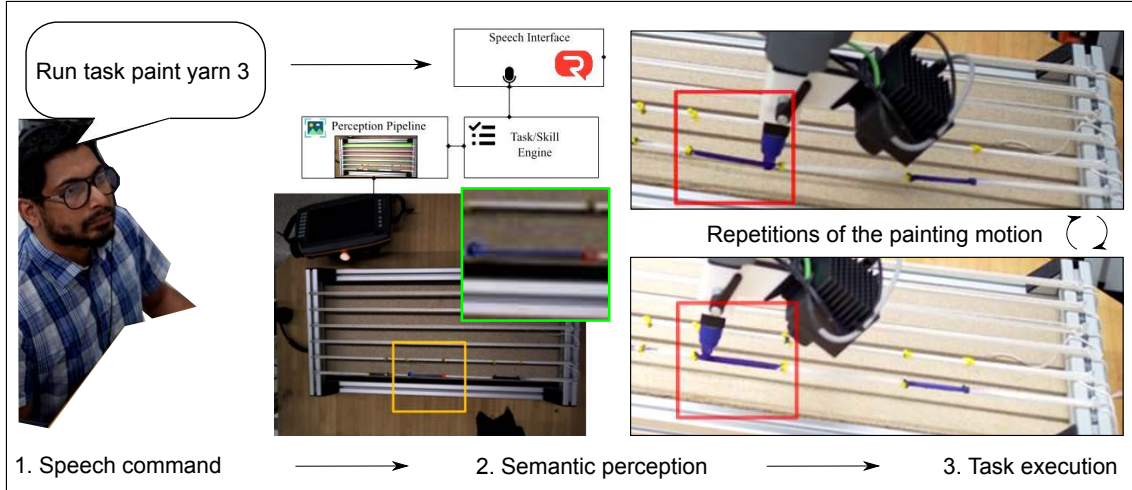


**Figure 10.1.:** Left: visualization of ikat, which is a preparatory phase for traditional textile manufacturing in Asia. The resulting fabric contains many patterns, which is marked and dyed by a craft person. Right: the system Yantra at Hannover Messe. Yantra supports the craftsmen by preparing the hand-loom and mark or paint the yarn threads with a collaborative robot.

creative tasks such as designing the new patterns for fabrics (see figure 10.1).

Yantra is a full stack robotics solution that comprises several modules from different robotic disciplines. At the hardware level, Yantra is based on an industrial collaborative robot called KUKA LBR iiwa 15 R930. Since ikat involves both painting and weaving tasks, the system is equipped with an adaptive tool-chain for different manipulation tasks. At the software level, we adapt a two-layer robot architecture with a top-down hierarchy. At the higher level of the architecture, the system is made of a language-to-motion framework, combining a speech interface, a large language model, and a skill engine. Given a speech command from a craftsman, the language model categorizes the given command into feasible actions and its parameters. The skill engine then generates concrete actions that are passed to the low level of the architecture. The reason for integrating language is to make the system easy to use. Craftsmen are not programmers and their hands might be tied while collaborating with a robot. At the low level of the architecture, the system contains the perception-action loop that executes the tasks given by the skill engine. The development of Yantra's vision system is based on the contributions of this thesis, which we detail below. Action generation is based on a cartesian impedance controller and an interpolator. Iiwa is a torque controlled robot with joint torque sensing capabilities.

Semantic perception of yarn threads is the primary requirement on Yantra's vision system. For example, the painting and the marking of the yarn threads require spatial understanding of the start and the end point of the yarn thread. As we base our developments on a speech interface, semantic understanding is also desired. For example, if the craftsman commands "paint yarn number three", the vision system needs to locate the yarn number three, which is a semantic perception task. In essence, the problem is semantic perception of linear deformable objects. To meet the aforementioned requirements, we decompose the given perception tasks into (a) semantic segmentation of yarn threads and (b) 3D localization of the start and the end point of the yarn thread. For the former, we utilize the tuft perception pipeline from chapter 7. Tufts are ropes and wires, which are another class of deformable linear objects. We use the depth information from a stereo camera for the 3D localization of the start and the end point. Obtained masks from semantic segmentation are used to extract two 3D points, which lie on both ends of the segmentation masks. In these ways,



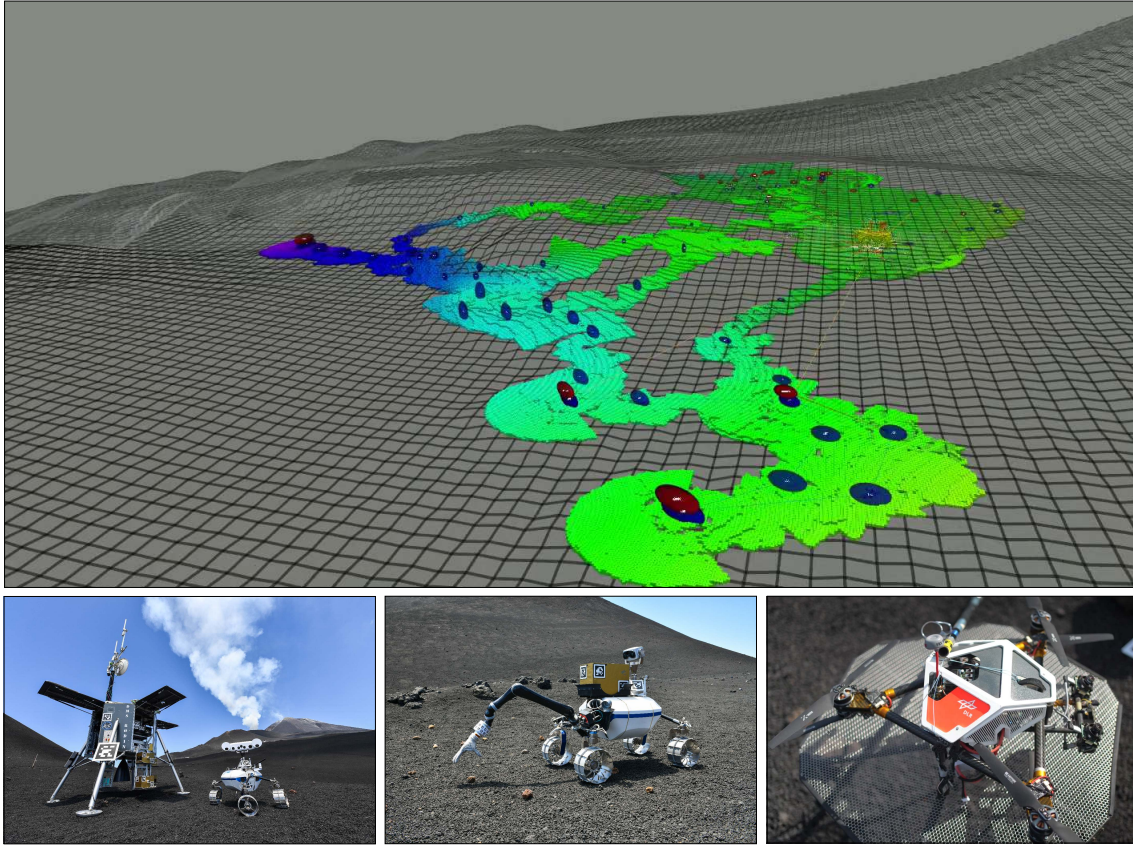
**Figure 10.2.:** Yantra performs semantic perception of yarn threads. With speech command from a craftsman, the robot’s skill engine queries the start (red) and the end point (blue) of the thread to be painted. Then, with the 3D localization of these points, the painting tasks are executed.

we transferred the results from chapter 7 to future manufacturing.

The main results are depicted in figure 10.2. We note that, similar to chapter 7, semantic segmentation is achieved without having to annotate the semantic masks for individual yarn threads. The achieved manipulation sequences project further the advantages of semantic perception for Yantra. In this example, a craftsman gives a speech command, “paint yarn number one” or “paint yarn number three”. Then, the system attempts to achieve the given manipulation tasks by first understanding where the targeted yarn thread is. After having the semantic segmentation masks related to the yarn thread, 3D localization of the start and the end points provides the robot’s motion control system with the reference locations to move. As a result, painting of the yarn thread is accomplished. We demonstrated Yantra at Hannover Messe 2024, where the planned demonstrations were successfully showcased to the public over five days. Yantra won the Finalist of the KUKA Innovation Award 2024.

## 10.2. The ARCHES Analog Space Demonstration Mission

At the DLR, we believe that heterogeneous teams of mobile robots will play an important role in future exploration of extraterrestrial planets. Heterogeneous teams of robots can complement each other in terms of mobility, sensing and manipulation capabilities, thereby enabling complex space missions for scientific discoveries and technology demonstrations. We have designed four different types of mobile and stationary assets, as the heterogeneous team of robots. These robots are shown in figure 10.3. The team consists of two rovers (one with a focus on optical scientific instruments and another one equipped with manipulation capabilities), one aerial system, and a lander. A micro aerial vehicle acts as a scout, because of its capabilities to fly from one point to another. The rovers may be limited in mobility, but they are less restrictive on payloads. Therefore, rovers carry scientific instruments and perform manipulation tasks. In the context of the ARCHES demonstration mission, we aimed to acquire several scientific data such as geological spectral imagery and laser-induced breakdown spectroscopy (LIBS) data, the collection of rock samples, and low-frequency ratio signals through a distributed antenna array. The mission was conducted during the four-week campaign at a Moon-analogue site on Mt. Etna, Sicily (Schuster et al., 2020).

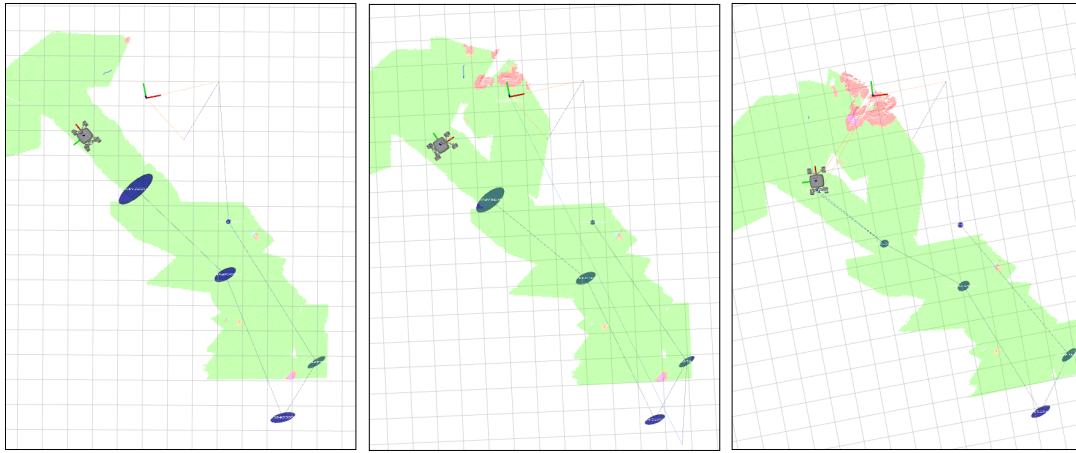


**Figure 10.3.:** Top: mapping of the test site on Mt. Etna, Sicily. Bottom: four robots are depicted. In sequence, a lander and a rover is on a Moon-analogue site, which is a volcanic mountain. Another rover equipped with a robotic manipulator is grasping a stone for sample collection. An aerial system, ARDEA acts as a scout. These robots form a heterogeneous team of mobile robots.

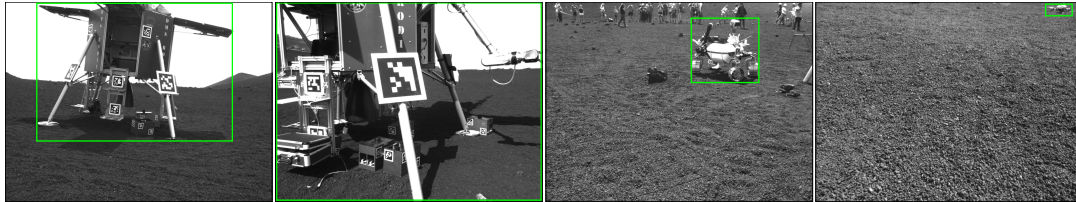
In planetary environments, no external means of global localization such as GNSS is available, and therefore, Simultaneous Localization and Mapping (SLAM) (Cadena et al., 2016) is one of the required competencies for the robots. Different robotic systems can create and share a common map. They are also aware of their locations in the given map, thereby enabling planning and coordination of cooperative multi-robot missions. Schuster et al. (2019) demonstrates a decentralized SLAM approach. In (Schuster et al., 2019), each robot uses stereo-inertial odometry based on an extended Kalman filter for local ego-motion estimation. Each robot also builds the so-called sub-maps, which are partial maps of limited size and high accuracy. Sub-maps, once completed, are then exchanged between the robots. Then, in a decentralized fashion, each robot builds a multi-robot pose graph for SLAM. The resulting factor graphs (Kschischang et al., 2002) are then optimized incrementally (Kaess et al., 2012), taking into account the sub-maps, keyframe matching, landmarks, and also the Kalman filter estimates. Re-localization and loop closures are based on a technique that uses Gaussian Processes (Giubilato et al., 2022), in order to embed a continuous representation of the gradients of the local terrain elevation. Unlike in man-made environments, planetary environments may not contain distinct landmarks. This characteristic challenges the existing re-localization and loop closure detection methods.

Inter-robot and intra-robot detections are our primary focus in this chapter. In the so-called collaborative SLAM, where different robots act as a team to build the map of

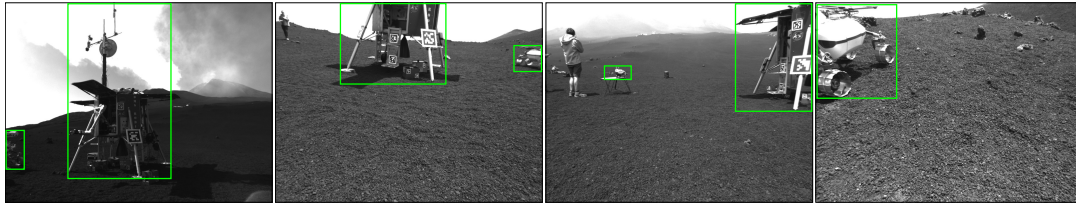




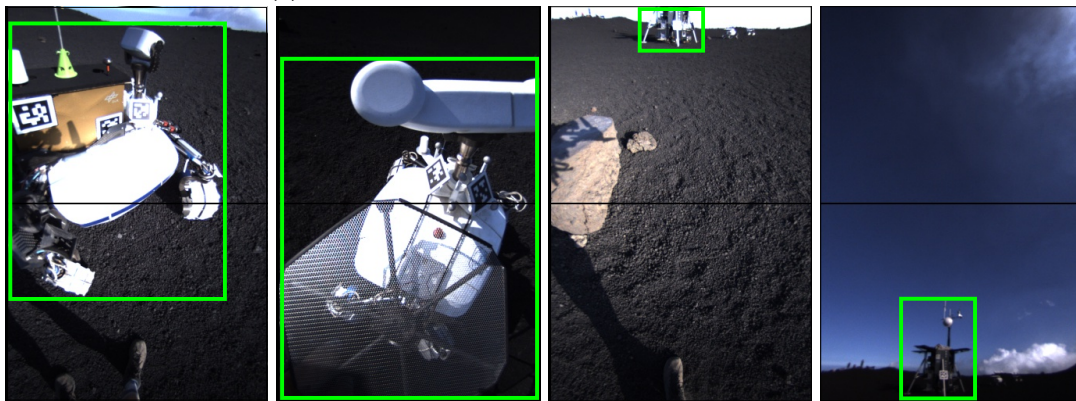
(a) Left: SLAM without robot detection as constraints. Middle: SLAM with bearing vector as constraints. Right: SLAM with AprilTag marker detection. Qualitatively, using bearing vector from an object detector resulted in similar results to SLAM with AprilTag marker in this example.



(b) Object detection results from the first rover.



(c) Object detection results from the second rover.



(d) Object detection results from the aerial system.

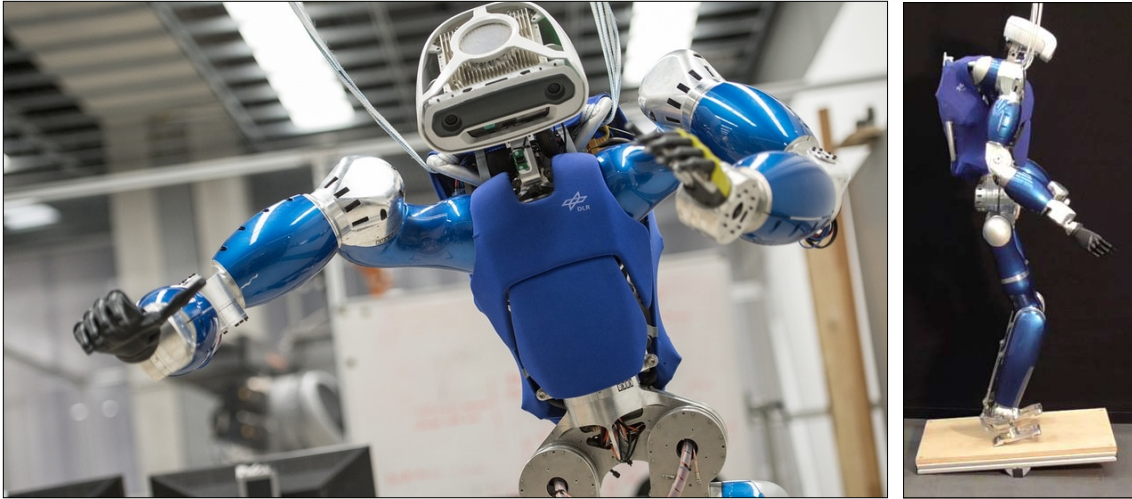
**Figure 10.4.:** Results of distributed SLAM and object detection.

the environments, an ability to detect the other robots belonging to the team can be useful (Schuster et al., 2019). These inter-robot and intra-robot detections, along with their relative locations, can be used within the SLAM graph as a pose constraint. When solving incremental optimization, such constraints can help in joining the maps from the respective robots into one coherent representation. During the ARCHES mission (Schuster et al., 2020), AprilTag markers (Wang & Olson, 2016) have been used for inter-robot and intra-robot detections. These so-called fiducial markers contain artificial features that can be decoded by the vision algorithms, allowing accurate 6D pose estimation of a planar surface. In the considered applications, the use of marker-based 6D pose estimation is well justified because robots and the lander can be accessible a-priori to the mission. However, two drawbacks should be noted. First, reflections can cause difficulties for the algorithm. This is because AprilTag detectors assume an estimate of a planar surface with four corner detections. Corners may not be visible depending on lighting conditions. Moreover, for similar reasons, markers that are far away from the robot may not be detected. To this end, we utilize a probabilistic object detector from chapter 5 for inter-robot and intra-robot detections. From 2D detections, bearing vectors are estimated to constrain the SLAM graph, enabling far-range detection of other robots in the team with uncertainty awareness.

The first results of our investigations are presented in figure 10.4. Note that all the results are from a collected multi-robot dataset from the ARCHES mission. To illustrate the problem, we also depict certain example images where AprilTag detectors have failed. Then, for far-range detections, we depict the results from the probabilistic object detector. Probabilities along with labels as well as covariances of the bounding boxes are displayed. These qualitative results show how a probabilistic object detector can be used for far-range, markerless inter-robot and intra-robot detections. Here, for training the detectors, a ratio of 0.8 is used for train and test set splits. Images from two rovers and the aerial system were annotated manually. Three detectors have been trained for each robot. Lastly, qualitative results on SLAM performance are also shown. In this example, an AprilTag on the lander is detected by a rover and as a result, the SLAM graph is constrained and incrementally optimized. Within the same scenario, we disable the AprilTag detector and compare the results with and without the probabilistic object detector. The results show that the SLAM results with the probabilistic object detector resemble more the results with the AprilTag detector. These results show how certain results of this thesis can be applied for SLAM.

### 10.3. Perceptive Locomotion with the DLR TORO

Next, we present the development of a vision system for locomotion of a humanoid robot. The developments are based on the DLR's torque-controlled humanoid robot, TORO (Engelsberger et al., 2014). TORO has 27 degrees of freedom with a height of 1.74m. The robot weighs about 79.2kg. Figure 10.5 shows the robot's hardware. With an RGB-D camera mounted on the head, two vision-based solutions have been demonstrated in the past. First, Kheddar et al. (2019) presents visual tracking with TORO in the context of aircraft manufacturing. In the final demonstration, TORO visually tracked a bracket as well as edges and key points of the fuselage. A classical key-point matching method was used to initialize the tracker. The used model-based tracker utilizes low-level edges extracted in the image, depth information, and points of interest (Trinh et al., 2018). These classical methods have been used for grasping and manipulation. The second vision-based solution was for locomotion (Mesesan et al., 2025). In Mesesan et al. (2025), AprilTag markers (Wang & Olson, 2016) were placed in the environments. Unlike applications in planetary exploration,



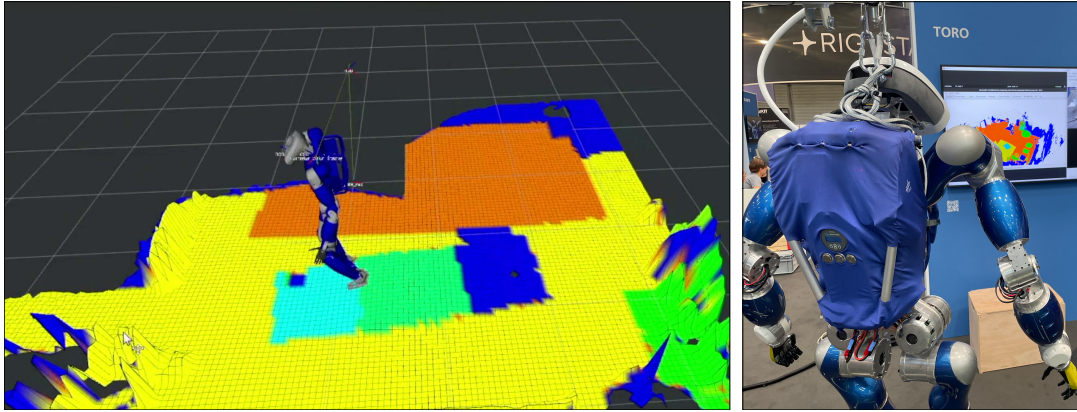
**Figure 10.5.:** Left: an upper body of TORO with its vision system is illustrated. Right: TORO is performing balancing task, illustrating its full body with legs.

the availability of the markers has been assumed in lieu of sophisticated perception. In this way, stairs could be localized in order to test the developed motion planner with TORO.

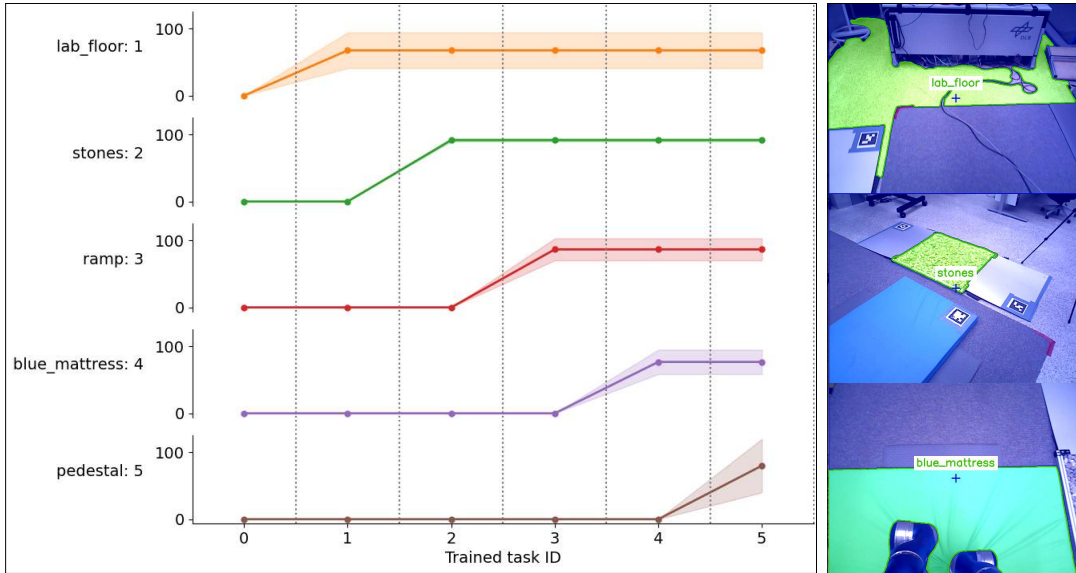
Advancing over these previous works, TORO’s new vision system has been designed to fulfill two desiderata while addressing the associated challenges. First, a marker-less solution to the navigation problems is desired. This is because no AprilTag markers can be readily assumed in real-world applications. The second requirement is the facilitation of adaptive locomotion strategies through semantic scene analysis of terrains. For example, on flat terrain, TORO may want to walk faster. If a difficult type of terrain lies ahead, say wet sands, TORO may decide to find another route towards its goal. To achieve the aforementioned requirements, several challenges can be foreseen. For the marker-less navigation, a SLAM system can be integrated for both state estimation and creating a metric map of the environments. In this case, as humanoid robots may fall due to wrong information from the vision system, robustness and reliability become key facets. Imagine TORO planning its next footstep wrongly due to drifts in its state estimation. For the semantic scene analysis of terrains, a challenge is distributional shifts that may fail the underlying deep learning-based perception. Here, one can imagine a deep learning model trained to classify grasses and carpets, but suddenly encounters bricks and stairs.

We developed a semantic SLAM system, incorporating a stream-based active learner from chapter 6. For state estimation, we combine robot pose estimates of ORB SLAM3 (Campos et al., 2021) and kinematics of TORO (Mesesan et al., 2025). TORO’s implementation computes the robot’s base frame and its velocity by utilizing the measurements of the joint positions, force/torque sensor, inertial measurement unit (IMU), and the robot’s kinematic model. The used fusion enables smooth and locally stable estimates of the robot’s state. For the semantic mapping, 2.5D grip map (Fankhauser & Hutter, 2016) has been extended. The mapping approach assumes a closed set of classes and fuses class probabilities over time for consecutive measurements. A low-pass filter is also integrated for reducing flickering and fluctuations of class labels. 2.5D map is integrated as an efficient representation for the applications in locomotion (Fankhauser & Hutter, 2016). The mapping module takes semantic segmentation as input, which is provided by an extension of CLEVER (chapter 6). In the considered use case, the robot only needs to know semantic information on its next





(a) Semantic SLAM from TORO's vision system. Left: semantic map of the lab environment. Right: semantic map and TORO's hardware at an industrial exhibition.



(b) Left: results from the stream-based active learner in sequentially learning new terrains. Right: few examples of semantic segmentation on considered terrain types.



(c) Left: TORO with a new vision system. Right: our user interface for stream-based active learning.

**Figure 10.6.:** Results of vision system development with TORO.



footsteps. Hence, SAM2 with a point prompt is used for obtaining segmentation masks. For human-robot interaction, we designed an extended reality app where the users can insert information about the terrain, instead of using the speech interface from chapter 6.

Results are shown in figure 10.6. First, semantic maps are displayed as qualitative results. The system creates the map at an interactive rate. The map captures different terrains using different colors. Output from the state estimation is used to further visualize the robot within the map. Second, quantitative results from CLEVER are depicted, which show the life-long learning aspect of the system without catastrophic forgetting. Examples of various terrains are also included for a better understanding of the results. With this, we show the robustness to distributional shifts for the applications in semantic SLAM, where the robot asked for help and learned online without catastrophic forgetting. Our new user interface is also displayed, which is based on an extended reality device called Microsoft HoloLens 2. TORO’s vision system was integrated into the live demonstrations at Automatica 2025, which is one of the largest industrial exhibitions in robotics and automation. Along with dynamic balancing and action generation using large language models, we demonstrated semantic SLAM with active learning. Figure 10.6 shows illustrations from the event.

## 10.4. Paula: Post-hoc Probabilistic Programming

Now, we present our concept behind a probabilistic programming library, Paula, being developed for uncertainty-aware computer vision in robotic applications.

Several definitions exist for the term *probabilistic programming* (Van De Meent et al., 2018). Some argue that a probabilistic programming language is an ordinary tool with random variable (`rand`) and related functions that support its users to understand the statistical behavior of the code. Others argue that probabilistic programming is a software tool that automatizes probabilistic inference, thereby promising to democratize probabilistic modeling. Both of these definitions merely emphasize different angles on the same idea. A probabilistic programming language supports random variables and probability distributions. Conditioning and marginalization are directly embedded in the program. The users can define a probabilistic model through a prior distribution and a posterior distribution. We take the latter definition, emphasizing the automatization of probabilistic inference.

For example, Stan (Carpenter et al., 2017) is arguably one of the most well-known probabilistic programming languages. In essence, Stan is a language that automatically computes derivatives and posterior probabilities for joint distributions. Consider a joint distribution  $p(\theta, y) = p(y|\theta)p(\theta) \propto p(\theta|y)$ . Once specified, Stan takes the data  $y$  as input and outputs an approximate sample from the posterior distribution. Diagnosis of the resulting inference procedures is also included in the program. To achieve these functionalities, many of the widely known probabilistic inference techniques are supported in the language. These include Markov Chain Monte Carlo (MCMC) sampling (Bishop & Nasrabadi, 2006) and its variants such as Hamiltonian Monte Carlo (Duane et al., 1987) and No-U-Turn sampler (Hoffman et al., 2014). Variational inference algorithms (Bishop & Nasrabadi, 2006) as well as Laplace Approximation are supported by the language. Stan supports Python and C++.

In probabilistic programming research, many recent endeavors focus on bringing differentiability into the programming language. A key idea behind differentiable programming such as PyTorch (Paszke et al., 2019), Tensorflow (Abadi et al., 2016) and JAX (Bradbury et al., 2018) is automatic differentiation, empowered by modern GPUs for parallel computing. Combining automatic differentiation with probabilistic programming brings an advantage of improving scalability with respect to large amounts of data and deep

learning architecture. Probabilistic techniques such as MCMC sampling and variational inference tend to be computationally expensive. Therefore, the use of GPU computing may enhance their efficiency, given that these probabilistic techniques often rely on differentiable quantities such as gradients and Hessians. This research direction resulted in many widely used libraries, namely Edward (Tran et al., 2017), pyprob (Baydin et al., 2019), NumPyro (Bingham et al., 2019), PyMC3 (Abril-Pla et al., 2023), tensorflow Probabilities (Abadi et al., 2016), probtorch (Paszke et al., 2019) and blackJAX (Bradbury et al., 2018).

#### 10.4.1. Why Post-hoc Probabilistic Programming?

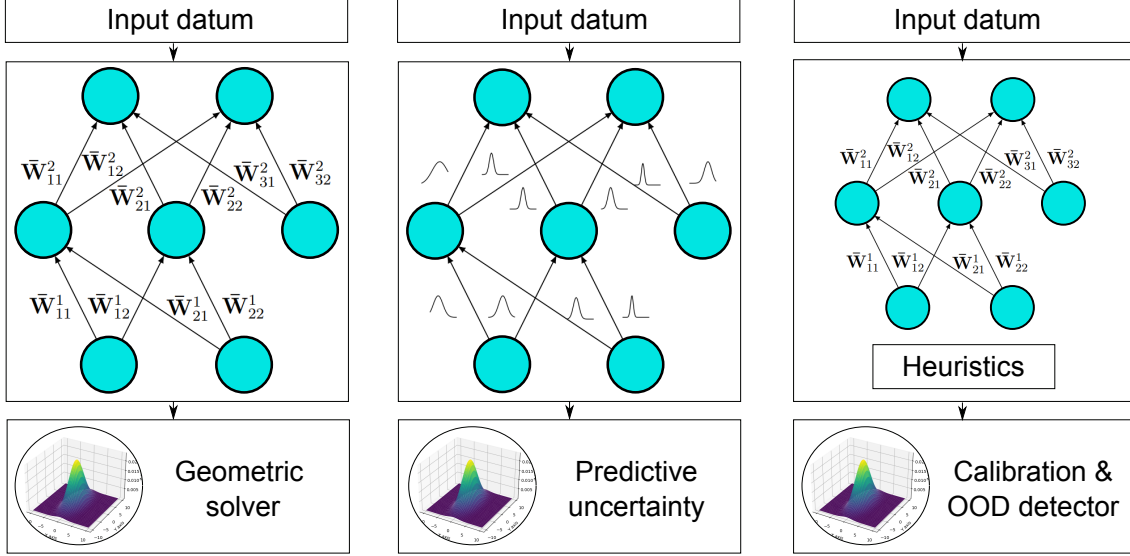
While the aforementioned advances provide useful tools to practitioners of probabilistic inference, their applicability might be limited to deep learning. To obtain a well-trained deep learning model, many heuristics such as batch normalization and non-standard loss functions are often used. As noted by Knoblauch et al. (2022), a direct application of variational inference can therefore be fragile, while MCMC sampling typically does not scale to large datasets and architectures (Wilson et al., 2022; Wenzel et al., 2020). Current state-of-the-art research is attempting to resolve these limitations by, for example, embedding stochasticity into the algorithms with a subset of data (Gawlikowski et al., 2023). However, these advances are not integrated into the current probabilistic programming languages.

Our main objective is to improve probabilistic programming so that the developed language can be more compatible for applications in computer vision and robotics. To do so, our key idea is to separate the modeling and the probabilistic inference in a post-hoc manner, hence coining the term post-hoc probabilistic programming. What do we exactly mean by this separation? This means we should design a probabilistic programming language, which takes an already well-trained deep learning model as an input, instead of training neural networks through probabilistic inference. Then, the language should be able to perform probabilistic inference on an already well-trained deep learning model in a post-hoc manner. In this way, practitioners of deep learning can build powerful models as they currently do, while probabilistic inference does not hurt the performance of existing models.

As an alternative, uncertainty quantification libraries are being built for addressing general applications of deep learning. Few notable examples are Fortuna (Detommaso et al., 2024), Lightning UQ Box (Lehmann et al., 2025) and Laplace (Daxberger et al., 2021a). These frameworks cover a variety of methods including Bayesian Neural Networks, evidential learning, and conformal predictions. However, in the context of probabilistic inference, these libraries lack the flexibility of probabilistic programming languages, as they mainly provide implementations of existing methods. As a result, these libraries do not cover broader use cases, especially for computer vision and robotic applications (see figure 10.7). For example, tasks within Lightning UQ Box are categorized as classification, 1-D and 2-D regression, and segmentation (Lehmann et al., 2025). Paula is also an uncertainty quantification library with a goal of increased compatibility for computer vision and robotics.

#### 10.4.2. Concept Description and Key APIs

To ensure compatibility of deep learning from computer vision, we categorize existing architectures into three categories, namely (a) geometric architectures, (b) end-to-end architectures, and (c) anomalous architectures. First, geometric architectures refer to methods that leverage deep learning to learn representations but use geometries to obtain the final outcome. One popular example is (He et al., 2022; Sarlin et al., 2021), where deep learning representations process the input images and Perspective-n-Point (PnP) solvers



**Figure 10.7.:** Left: An architecture where geometric solvers are used to derive the final prediction in a computer vision pipeline. For this type of pipelines, Paula performs probabilistic inference on geometric solvers. Middle: An end-to-end architecture that makes predictions standalone. Paula performs probabilistic inference on deep learning model for uncertainty quantification. Right: An architecture where heuristics are used to derive the final prediction. For such cases, calibration methods and Out-of-Distribution (OOD) detectors can be combined using Paula.

derive the 6D poses. Second, end-to-end architectures refer to computer vision methods where deep learning learns the mapping from input to the desired output directly. A typical example is classifiers such as ResNet (He et al., 2016). Lastly, anomalous architectures are methods that do not fall into the previous categories. We refer to such cases as anomalous architectures, often characterized by heuristics. A popular example is YOLO (Redmon et al., 2016), where non-maximum suppression is applied to the output of the deep learning models. Below, we lay out our key APIs that address each of these categories.

- **Probabilistic inference in geometric architectures.** In this class of methods, geometric algorithms in computer vision are used at the final stage of the overall pipeline. To quantify uncertainties of the overall pipeline, Paula performs probabilistic inference on geometric algorithms. In this way, we avoid probabilistic inference on deep learning models directly. For solvers of the geometric problems such as point cloud registration and the Perspective-n-Point problem, both sampling and variational inference techniques have been investigated (Maken et al., 2021, 2022; Vakhitov et al., 2021). Paula supports some of these solvers with probabilistic inference. Extensions to multi-view geometry are also envisioned.

- **Probabilistic inference in end-to-end architectures.** For this category of methods, we directly apply probabilistic reasoning to the deep learning model. As discussed throughout the thesis, this procedure involves the prior specification, the posterior inference, and the marginalization for predictive uncertainty estimation. Using Laplace Approximation and its variants, we can obtain the posterior distribution over the model in a post-hoc manner, i.e., maximum a-posteriori estimates are obtained through the standard training pipeline. Then, Laplace Approximation computes the covariance matrix without changing the already well-trained neural networks. The contributions of this thesis are integrated into Paula.

- **Probabilistic inference in anomalous architectures.** This class of methods is characterized by heuristics, in which uncertainty propagation is non-trivial. In non-minimum

suppression, for example, no gradients may exist, and even the first-order uncertainty propagation might be difficult. In this case, confidence calibration can be applied for both regression and classification tasks (Gawlikowski et al., 2023). Post-hoc calibration methods (Guo et al., 2017) typically take the output of a pipeline and use held-out validation data to learn a mapping that reduces the chosen calibration metric. Then, out-of-distribution detectors (Feng et al., 2023), based on generative models, can be combined in order to identify distributional shifts. Including conformal predictions is also envisioned.

With these key APIs, Paula introduces a new concept of post-hoc probabilistic programming. Paula’s users can first train a deep learning model in a standard fashion. Then, without changing the already well-trained neural networks, probabilistic inference can be performed and uncertainty estimates can be obtained. Instead of examining the domain-specific problem at hand individually, Paula brings a plug-in and plug-out philosophy by separating the representation learning and the probabilistic inference. Flexibility is achieved through a new categorization of existing computer vision architectures.

## 10.5. Summary

Several applications of our probabilistic methods to the on-going DLR projects are presented in this chapter. Future manufacturing with Yantra, planetary exploration with the ARCHES mission, and perceptive locomotion with the DLR TORO are concretely discussed. These projects show the broader applicability of our contributions in robotics. Then, core concept behind a library called Paula is proposed, which automatizes probabilistic inference in deep learning-based computer vision. Many exciting future venues of research are envisioned. For example, integration of object detectors and 6D pose estimators into distributed SLAM can increase the performance of mobile robots for future planetary missions. Industry is currently looking into the applications of humanoid robots in manufacturing facilities. Optimus from Tesla and Atlas from Boston Dynamics are notable examples. Our development activities on TORO can also be a useful reference that accelerates the on-going progress.

## 11.1. Contributions

For more than a decade, deep learning has been making encouraging progress in many disciplines such as computer vision and natural language processing. However, applications of deep learning in robotics motivate certain research questions that typically differ from those of computer vision and natural language processing. For example, a deep learning method may misclassify an image or a large language model such as chatGPT may produce hallucinated answers. In computer vision and natural language processing, all these mistakes have no physical consequences. On the other hand, robots are physical agents that act and interact with the real world. Their mistakes can lead to potentially catastrophic results. Imagine accidents caused by an autonomous car or a surgical robot. Therefore, such facets of robotics research motivation further open questions: How much can we trust the predictions of deep learning methods when the failures have catastrophic consequences? For this, can we estimate uncertainties associated with their predictions?

This thesis investigates the problem of uncertainty estimation in deep learning. For the given problem, Bayesian statistics have been widely studied over many decades. Relying on Bayesian statistics provides a principled way of incorporating domain knowledge through priors, quantifying uncertainties about the models, and making predictions through marginalization, potentially improving the accuracy and generalization. However, in practice, priors may be misspecified while posteriors can only be crudely approximated, leading to poorly calibrated uncertainty estimates. Predictions through marginalization can also be computationally expensive, prohibiting their usage in real-time applications with constraints. For these reasons and many more, arguably, applications of Bayesian statistics to deep learning have been limited in robotics, e.g., no publicized deployments of these methods in robotic industrial practices, despite their promises and theoretical advantages.

The main contributions of this thesis are two-fold. On the fundamental side, we proposed Bayesian methods to quantify uncertainty in deep learning. We advance the state-of-the-art Bayesian methods in the relevant metrics for robotics while providing theoretic foundations behind the proposed methodologies. On the applied side, we have developed probabilistic systems based on the proposed methods. At the system level, we show that these methods can be utilized to improve the reliability and robustness of robotic systems, or address

	On prior	On posterior	On prediction
<b>Research gap</b>	Existing works choose uninformative prior despite the signs of prior misspecification.	Existing works simplify the covariance matrix for scalability in lieu of expressivity.	Existing works require combining multiple predictions from the model’s distribution.
<b>Major contributions</b>	A novel method for learning scalable and structured posteriors of neural networks as information priors with generalization theory.	A novel information theoretic formulation that exploits the sparsity of inverse space of posteriors and its inference procedures.	A novel neural tangent kernel theory for sampling-free uncertainty estimation with mixtures of Gaussian Process experts.
<b>Ramification</b>	Domain knowledge can be incorporated for generalization and uncertainty estimation.	More expressive but memory efficient approximation of posterior can be obtained.	Real-time uncertainty estimation becomes possible with Gaussian Process formulation.
<b>Real-world demonstrations</b>	Lead to a stream-based active learning system for robust semantic perception from human instructions.	Lead to a system for fluid flow visualization and an introspective perception system for aerial manipulation.	Lead to a shared autonomy system that enables robotic manipulation under uncertainty in deep learning.
<b>Preserving accuracy</b>	No design guarantees on preservation of accuracy (a-priori method).	Preserves accuracy of deep learning by design (post-hoc method).	Preserves accuracy of deep learning by design (post-hoc method).
<b>Memory efficiency</b>	Deployable on embedded GPUs. Memory grows with tasks.	Deployable on embedded GPUs. Most memory efficient.	Deployable on embedded GPUs. Memory grows with data.
<b>Run-time efficiency</b>	Requires sampling for predictions and efficient only when parallelized.	Requires sampling for predictions and efficient only when parallelized.	Does not require sampling and thus efficient without parallelization.
<b>Demonstrated scalability</b>	ImageNet-1K data-set within 24 hours on NVIDIA 1080 GPU.	ImageNet-1K data-set within 24 hours on NVIDIA 1080 GPU.	Two million data points within 15 hours on NVIDIA 1080 GPU.
<b>Use cases</b>	Excels in small data regime where the prior term dominates over the likelihood.	Excels in large data regime, large architectures but requires small memory consumption.	Excels in data regime upto 2 million. Permits large input but smaller output space.
<b>Publications</b>	<a href="#">Schnaus et al. (2023)</a> ; <a href="#">Lee et al. (2025b)</a> .	<a href="#">Lee et al. (2020b, 2024, 2020a, 2023)</a> .	<a href="#">Lee et al. (2021, 2025a)</a>

**Table 11.1.:** A probabilistic robotics perspective to uncertainty in deep learning. This thesis addresses one problem: uncertainty quantification in deep learning. Contributions are summarized and also compared among each other. Use cases for each methods are also reported.

specific problems in real-world applications. In light of the aforementioned research gaps, we address the problem of (a) the prior specification to incorporate domain knowledge, (b) scalable inference of the posteriors to represent model uncertainty, and (c) efficient marginalization procedures to enable real-time uncertainty quantification in robotic systems.

On meaningful priors, we propose to learn an informative prior, which can then be used in subsequent tasks to learn Bayesian neural networks. The proposed method uses Laplace Approximation to treat the posteriors of the previous tasks as the prior for the current task, which is combined with a derived optimization technique to upper bound the generalization error using PAC-Bayes theory. In this way, the prior can be used to improve uncertainty estimates and learn the predictors with tighter generalization bounds. The key enablers are our technical contributions, namely the derivation of the sums-of-Kronecker-product computations that provably converge to the rank-one optimality, and PAC-Bayes objectives that produce tractable generalization bounds. Among many experiments, we demonstrated the proposed methods on a humanoid robot to improve the robustness of semantic perception. Here, uncertainty estimates are used to ask for help from humans, and the prior enables the online adaptation of the model for robustness in semantic perception.

On scalable posteriors, we propose a sparse formulation that relies on an inverse formulation of the Multivariate Normal Distributions, also known as the information form. The main findings are that this formulation has two important ramifications on developing scalable solutions to the given problem. Firstly, its inference can rely on Laplace Approximation, which involves scalable computations of the Kronecker-factored approximations of the information matrix (i.e. the inverse of the covariance matrix). Moreover, as the information matrix tends to be sparse in its spectrum within neural networks, sparse formulations can be naturally exploited for improving space complexity. To do so, a low-rank Kronecker-factored eigen-decomposition is derived with the proposed spectral sparsification algorithm that preserves the Kronecker products in its eigen-basis. Exploiting the scalability, we showcase pool-based active learning in two computer vision applications, namely fluid flow visualizations and introspective perception of an aerial manipulation system. We show improvements in sample efficiency of labeled data for these concrete applications.

On efficient predictions, we propose a method that provides uncertainty estimates based on a mixture of Gaussian Process (GP) experts, which is combined with a neural network so that the resulting predictor is also accurate. As a theoretic motivation, proof is provided to show that neural networks can be cast as a mixture of GP experts with a Neural Tangent Kernel. To bring the derived theory into practice, a divide-and-conquer strategy and an efficient prediction algorithm are devised. In the division step, the input space is clustered into local areas, one for each GP expert, and the discontinuity that occurs between neighboring GPs is further resolved in the conquer step. At test time, the resulting predictive model is real-time capable and maintains the predictive power of deep learning. We exploit the run-time efficiency for developing a shared autonomy system for aerial manipulation. The key idea is to adapt the level of autonomy using uncertainty estimates from the robot’s perception. Ample evidence is provided to demonstrate that uncertainty estimates enable safe integration of deep learning-based perception in robotic systems.

The main contributions of this thesis are summarized in table 11.1. The three proposed methodologies, albeit focused on different aspects of Bayesian statistics, address one problem: uncertainty in deep learning. Hence, we report the comparisons of these methods and lessons learned from our experiences of extending them towards real-world applications. In that respect, the contributions of this thesis stand on two pillars of probabilistic robotics – one on fundamental algorithmic work for uncertainty, and another on the development of



novel probabilistic systems. Through this lens of probabilistic robotics, we provide several encouraging advances on the problem of uncertainty in deep learning, paving the way towards safe, robust, and trustworthy robotic systems with modern artificial intelligence.

## 11.2. Outlook

While we contribute several novel methods and showcase their implementations on real robots and applications, many open questions remain for further improvements and extensions. Next, we outline four potential extensions of this thesis as future work.

**Bayesian foundational models.** Foundational models powered by generative models are gaining popularity in robotics. I believe that Bayesian statistics and foundational model research can mutually benefit each other. In one direction, foundational models also require uncertainty estimates. This may resonate, given many recent attempts (Zitkovich et al., 2023) to directly map speech and visual input to action commands. In the other direction, the availability of data and architectures that broadly generalize over different tasks can benefit Bayesian inference. Recapping our work on priors, the informative priors assume relevant data and models to learn from. We might be able to obtain foundational priors that generalize and help Bayesian inference over many probabilistic reasoning tasks.

**Expressive probabilistic models.** What might be the most powerful method in the landscape of uncertainty quantification for deep learning? Many years ago, a competition (Wilson et al., 2022) was held in order to seek this question. The competition attracted participation from several machine learning research groups around the world. The top five winning methods were all based on a combination of deep ensembles with variations in Bayesian methods – the two popular approaches that were often thought to be competing in the computer vision community. Many attributed the success of these hybrid methods to the expressivity of probability distributions, i.e., deep ensembles capture different modes of the posteriors while Bayesian methods capture probability mass around the modes (Wilson et al., 2022). However, bringing these methods to robotics requires significant improvements in computational efficiency, which can be an exciting direction for further research.

**Scalable function-space models.** One of the exciting ongoing debates is on weight-space versus function-space formulations. Weight-space formulations model priors and posterior distributions over the weights. In contrast, function-space formulations consider the distribution over functions a model can induce. A canonical example is GPs. Function-space formulation allows reasoning about uncertainty in predictions directly but might be computationally intractable. In this thesis, we have explored both formulations. We believe that enhancing the scalability of function-space formulations can be a promising direction. When utilizing GPs, Wilson et al. (2025); He et al. (2020) provide a theory that sampling near the loss landscape is equivalent to drawing samples in function-space. Such methods may significantly address the computational problems in GPs. Function-space Bayesian Neural Networks (Sun et al., 2019; Rudner et al., 2022a) can also provide an alternative.

**Representation learning matters.** A key design choice we made was to build on post-hoc methods, i.e., probabilistic inference techniques that are applied after a deep learning model is trained, unlike variational inference or sampling. The advantage of post-hoc methods is the compatibility with the modern deep learning arsenal of complex training pipelines and loss functions. This means we can directly work with pre-trained models and reason about uncertainty. The predictive accuracy of the model remains unchanged by design. This design choice, however, might be suboptimal. For example, Laplace Approximation is limited to bell-shaped distributions. Many existing works show that

indeed, representation learning matters for uncertainty quantification ([Liu et al., 2020b](#)). Yet, these methods are known to be fragile ([Knoblauch et al., 2022](#)). Here, advancing post-Bayesian methods such as Nightingale posteriors and generalized variation inference can be exciting future venues for both machine learning and probabilistic robotics.



They say “theory informs practice” and “practice validates and refines theory”. It is my hope that both theoretical and practical contributions of this thesis jointly enable a deeper understanding of the priors, the posteriors, and the predictions in artificial intelligence.



# Part V.

## Supplementary Materials

“Details matter. It’s worth waiting to get it right” – Steve Jobs.



## A.1. Preliminaries

Here, we first give an overview of the concept on Fisher Information Matrix of Neural Networks. Moreover, we discuss the main idea of the PAC-Bayesian theory.

### A.1.1. Fisher Information Matrix

**Fisher information matrix.** As  $\hat{\theta}$  could be a saddle point or due to numerical instabilities, the Hessian could be indefinite, in which case the normal distribution is not well defined (Martens, 2014; Botev et al., 2017). Therefore, positive semidefinite approximations of the Hessian are used like the *Fisher Information Matrix* (or the Fisher matrix) or *Gauss-Newton matrix*. For a likelihood of an exponential family, categorical and normal distributions, both approximations are the same, and for piecewise linear activation functions like ReLU (Nair & Hinton, 2010), they moreover coincide with the Hessian (Martens & Grosse, 2015).

**Definition A.1.1** (Fisher Information Matrix (Martens & Grosse, 2015)). Let  $P$  be a distribution over  $\mathcal{X} \times \mathcal{Y}$  and  $p(\cdot|\mathbf{x}, f_{\theta})$  be a conditional distribution over  $\mathcal{Y}$  dependent on the parameter vector  $\theta$ . Then the Fisher matrix is defined as

$$\mathbf{F} = \mathbb{E}_{(\mathbf{x}, \_) \sim P} \mathbb{E}_{\mathbf{y} \sim p(\cdot|\mathbf{x}, f_{\theta})} \left[ \frac{d \ln p(\mathbf{y}|\mathbf{x}, f_{\theta})}{d\theta} \frac{d \ln p(\mathbf{y}|\mathbf{x}, f_{\theta})}{d\theta}^T \right].$$

Here, the underscore indicates that the corresponding variable is not used.

*Remark A.1.2.* Note that the targets from the training data are not used to compute the Fisher matrix. Using the ground truth targets instead of samples from the predictive model distribution would lead to the empirical Fisher matrix.

For an easier notation, we write  $\mathbb{E}$  instead of  $\mathbb{E}_{(\mathbf{x}, \_) \sim P} \mathbb{E}_{\mathbf{y} \sim p(\cdot|\mathbf{x}, f_{\theta})}$  and  $\mathcal{D} \cdot = \frac{d \ln p(\mathbf{y}|\mathbf{x}, f_{\theta})}{d \cdot}$  for the derivative of the log-likelihood in the following. Moreover, we drop the layer index for the activations and pre-activations *i.e.*  $\bar{\mathbf{a}} = \bar{\mathbf{a}}_l$  and  $\mathbf{s} = \mathbf{s}_l$ .

**The Fisher approximations.** The full Fisher matrix and even a block-diagonal approximation without correlations between different layers are usually not feasible to store or compute for modern neural networks (Martens & Grosse, 2015). Common approximations of the block-diagonal form are by a diagonal or a Kronecker-factored matrix. The Kronecker-factored approximation comes from the fact that for a single sample, the Fisher matrix is the sum of Kronecker-factored matrices. For fully connected layers, the derivative after the weight matrix can be computed as  $\mathcal{D}\mathbf{W} = \mathcal{D}\mathbf{s}(\bar{\mathbf{a}})^T$ . With this, the block of the Fisher matrix corresponding to layer  $l$  is given by  $\mathbf{F}_l = \mathbb{E}[\bar{\mathbf{a}}(\bar{\mathbf{a}})^T \otimes \mathcal{D}\mathbf{s}(\mathcal{D}\mathbf{s})^T]$ .

As the convolutional layers share the weight tensor among the spatial positions, the derivative is the sum of the outer products:  $\mathcal{D}\mathbf{W}_{i,k} = \sum_{\mathbf{t} \in \mathcal{T}} \mathcal{D}\mathbf{s}_{\mathbf{t},i} \bar{\mathbf{a}}_{k,\mathbf{t}}$ . Therefore, the Fisher matrix block for layer  $l$  can be formulated as  $\mathbf{F}_l = \mathbb{E}[\sum_{\mathbf{t} \in \mathcal{T}} \sum_{\mathbf{t}' \in \mathcal{T}} \bar{\mathbf{a}}_{\mathbf{t},\mathbf{l}} (\bar{\mathbf{a}}_{\mathbf{t}',\mathbf{l}})^T \otimes \mathcal{D}\mathbf{s}_{\mathbf{t}} (\mathcal{D}\mathbf{s}_{\mathbf{t}'}^T)]$ .

**Kronecker-Factored Approximate Curvature.** The *Kronecker-Factored Approximate Curvature* (KFAC) approximates this Fisher matrix of a fully-connected layer (Martens & Grosse, 2015) and a convolutional layer (Grosse & Martens, 2016) by

$$\mathbf{F}_l \approx \mathbb{E}[\mathbb{E}[\bar{\mathbf{a}}(\bar{\mathbf{a}})^T \otimes \mathcal{D}\mathbf{s}(\mathcal{D}\mathbf{s})^T]] \quad \text{and} \quad \mathbf{F}_l \approx \frac{1}{|\mathcal{T}|} \mathbb{E}[\bar{\mathbf{a}}(\bar{\mathbf{a}})^T] \otimes \mathbb{E}[(\mathcal{D}\mathbf{s})^T \mathcal{D}\mathbf{s}],$$

respectively. In particular, KFAC approximates the expected Kronecker product as a Kronecker product of expectations, which is not true in general but leads to Kronecker-factored blocks of the Fisher matrix. Kronecker factorization enables the storage of two smaller matrices rather than one large matrix (Martens & Grosse, 2015). However, these factorizations assume that the activations and the corresponding pre-activations are statistically independent, which is usually not met in practice (Tang et al., 2021). Furthermore, additional assumptions such as the independence of the first- and second-order statistics of the spatial positions are used for convolutional layers, which might impair its approximation quality.

**Kronecker-Factored Optimal Curvature.** Another tractable approximation of the Fisher matrix is the *Kronecker-factored Optimal Curvature* (KFOC) (Schnaus et al., 2021) which finds optimal Kronecker factors for each batch of data points and approximates both the linear and the convolutional layer as a Kronecker product with two factors,

$$\mathbf{F}_l \approx \mathbf{A}_l \otimes \mathbf{G}_l.$$

It transforms the problem of finding optimal Kronecker-factors into the best rank-1-problem and solves it with a scalable version of the power method. The approximation is usually closer to the Fisher matrix than the approximation by KFAC in terms of the Frobenius error.

**Laplace Approximation for KFAC and KFOC.** Given a block-diagonal Kronecker-factored precision (or information) matrix  $\mathbf{H} = \text{diag}(\mathbf{A}_1 \otimes \mathbf{G}_1, \mathbf{A}_2 \otimes \mathbf{G}_2, \dots, \mathbf{A}_L \otimes \mathbf{G}_L)$ , the normal distribution reduces to  $L$  independent matrix normal:

$$\mathcal{N}(\hat{\boldsymbol{\theta}}, \mathbf{F}^{-1}) = \prod_{l=1}^L \mathcal{MN}(\hat{\mathbf{W}}^l, \mathbf{A}_l, \mathbf{G}_l).$$

In the following, we assume a block-diagonal approximation of the Fisher matrix where each block  $\mathbf{F}_l \in \mathbb{R}^{n_l \times n_l}$  corresponds to a layer  $l \in [L]$ . For fully connected and convolutional layers, the dimensionality  $n_l$  is the size of the vectorized weight matrix, *i.e.* for fully connected layers  $n_l = d_l(d_{l-1} + 1)$  and for convolutional layers  $n_l = c_l(c_{l-1}|\Delta^l| + 1)$ .



### A.1.2. PAC-Bayesian Bounds

Even though Bayesian neural networks were introduced in the Bayesian framework, *Probably Approximately Correct (PAC)-Bayesian bounds* introduced a Frequentist method to bound the generalization of statistical functions like Bayesian neural networks (Germain et al., 2016). PAC bounds aim to upper bound the risk, *i.e.* the expected loss on the true data distribution, with high probability using properties of the architecture and optimization of neural networks (Wolf, 2018). PAC-Bayesian bounds obtain similar bounds for Bayesian neural networks by comparing the posterior distribution with a dataset-independent prior distribution (Guedj, 2019).

Let  $\mathcal{G} = \{g : \mathcal{X} \rightarrow \mathcal{Y} \mid g \text{ is measurable}\}$  be the set of hypotheses and a corresponding  $\sigma$ -algebra  $G$  such that  $(\mathcal{G}, \mathfrak{G})$  is a measurable space. Denote the set of probability distributions on this measurable space as

$$\mathcal{M} = \{\mu : \mathcal{G} \rightarrow [0, 1] \mid \mu \text{ is a probability distribution on } (\mathcal{G}, G)\}.$$

Moreover, let  $l : \mathcal{G} \times \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$  be a loss function. Then define the risk for  $g \in \mathcal{G}$  as

$$\mathcal{L}_P^l(g) = \mathbb{E}_{(\mathbf{x}, \mathbf{y}) \sim P} [l(g, \mathbf{x}, \mathbf{y})]$$

and its empirical counterpart on the training data as

$$\hat{\mathcal{L}}_D^l(g) = \frac{1}{N} \sum_{i=1}^N l(g, \mathbf{x}_i, \mathbf{y}_i).$$

Note that the neural network  $f_{\theta}$  as defined above is not in  $\mathcal{G}$  because its output does not have to be in  $\mathcal{Y}$ . It only specifies the parameters of a distribution  $p(\cdot | \mathbf{x}, f_{\theta})$  on  $\mathcal{Y}$ . Nonetheless, for example, the function defined by  $x \mapsto \arg \max_{y \in \mathcal{Y}} p(y | \mathbf{x}, f_{\theta})$  is in  $\mathcal{G}$ .

Given a data-set independent prior distribution over the hypothesis set  $\pi \in \mathcal{M}$ , the PAC-Bayesian theory bounds the probability that the expected risk is large for hypotheses sampled from another probability distribution  $\rho \in \mathcal{M}$  which is absolutely continuous w.r.t.  $\pi$ :

$$P_{D \sim P^N} (\forall \rho \in \mathcal{M}, \rho \ll \pi : \mathbb{E}_{g \sim \rho} [\mathcal{L}_P^l(g)] \leq \delta(\rho, \pi, \mathcal{D}, \varepsilon)) \geq 1 - \varepsilon, \quad (\text{A.1})$$

for  $\varepsilon > 0$  and the PAC-Bayesian bound  $\delta(\rho, \pi, \mathcal{D}, \varepsilon)$  (Guedj, 2019).

The first bound was introduced by McAllester (McAllester, 1999b,a, 2003b,a) for bounded loss functions.

**Definition A.1.3** (McAllester Bound (Guedj, 2019)). Let  $\varepsilon > 0$ ,  $\rho, \pi \in \mathcal{M}$ ,  $\rho \ll \pi$  and  $l : \mathcal{G} \times \mathcal{X} \times \mathcal{Y} \rightarrow [0, 1]$ , then

$$\delta(\rho, \pi, \mathcal{D}, \varepsilon) = \mathbb{E}_{g \sim \rho} [\hat{\mathcal{L}}_D^l(g)] + \sqrt{\frac{\mathbb{KL}(\rho \| \pi) + \ln \frac{2\sqrt{N}}{\varepsilon}}{2N}} \quad (\text{A.2})$$

is an upper bound for equation A.1.

The bound for the error loss function was improved by Catoni (2007) when  $\frac{\mathbb{KL}(\rho \| \pi)}{N}$  is large. The error loss function, or 0-1-loss, is defined as

$$\text{er} : \mathcal{G} \times \mathcal{X} \times \mathcal{Y} \rightarrow \{0, 1\}, \text{er}(g, \mathbf{x}, \mathbf{y}) = \mathbf{1}[f(\mathbf{x}) \neq \mathbf{y}], \quad (\text{A.3})$$

where  $\mathbf{1}[s]$  is one if the statement  $s$  is true and else zero.

**Definition A.1.4** (Catoni Bound (Catoni, 2007)). Let  $\varepsilon > 0$ ,  $\rho, \pi \in \mathcal{M}$  and  $\rho \ll \pi$ , then

$$\delta(\rho, \pi, \mathcal{D}, \varepsilon) = \inf_{c>0} \frac{1 - \exp(-c\mathbb{E}_{g \sim \rho}[\hat{\mathcal{L}}_{\mathcal{D}}^{\text{er}}(g)] - \frac{\mathbb{KL}(\rho \parallel \pi) - \ln \varepsilon}{N}}{1 - \exp(-c)} \quad (\text{A.4})$$

is an upper bound for equation A.1.

*Remark A.1.5.* Note that in the PAC-Bayesian literature,  $\rho$  is called posterior even though it is an arbitrary distribution that is dependent on the data (Guedj, 2019). To distinguish this from the posterior computed by Bayes’ rule, we will explicitly write PAC-Bayes posterior if we do not use Bayes’ rule.

The PAC-Bayes bounds depend mainly on two terms: the KL-divergence of the PAC-Bayes prior and posterior and the empirical risk. Therefore, these bounds are small when the posterior is close to the prior and the loss on the training data is low. Thus, a good model should explain the training data well while not relying too much on it. In PAC-Bayesian bounds, the prior and posterior are both distributions over the functions and not over the weights like in Bayesian neural networks. Hence, to apply the PAC-Bayesian bounds for Bayesian neural networks, one needs to identify each weight sample with a sample in the function-space. However, neural networks are not identifiable (Brea et al., 2019; Pourzanjani et al., 2017). Thus, multiple different weight vectors can explain the same function given by a neural network architecture. The KL-divergence is therefore smaller in the function-space than in the weight-space, and an upper bound is obtained by considering the distributions over the weights (Dziugaite & Roy, 2017).

In this work, we use the two bounds introduced above. Nevertheless, we find an approximate upper bound of the expected empirical error for the Laplace approximation and can compute the KL-divergence in closed form for our models. Hence, all results of this work can directly be applied to other PAC-Bayesian bounds given that they depend on the expected empirical error and the KL-divergence of the posterior and the prior.

## A.2. Algorithmic Overview, Complexity and Extensions

This section summarizes the training of BPNNs and their computational complexity. After that, we present the details about the proposed sums-of-Kronecker product computations. First, we review the basic idea of progressive neural networks (PNNs) (Rusu et al., 2016) and their extension to arbitrary network architectures such as ResNets (Rusu et al., 2016). We then describe the training process and present a pseudocode for it. Finally, we analyze the complexity of training BPNNs. We use the same setup as in section 3.2.4 by considering the tasks  $\mathfrak{T}_0, \dots, \mathfrak{T}_T$ , where  $\mathfrak{T}_0$  is used to learn the prior. Each task  $\mathfrak{T}_t$  contains a training dataset  $\mathcal{D}_t = \left( (\mathbf{x}_i^{(t)}, \mathbf{y}_i^{(t)}) \right)_{i=1}^{N_t}$ .

### A.2.1. Progressive Bayesian Neural Networks

**Implementation of PNNs.** Given a network architecture, PNNs create a copy of that network, called a column, for each new task and also add lateral connections between different columns to promote positive transfer between tasks. Lateral connections are parameterized functions that combine the features of previous layers with those of the current layer. Unlike Rusu et al. (2016), we use the same function for lateral connections as for the main column, instead of their adapter architecture. Let  $\mathbf{s}_l^{(t)}$  and  $\mathbf{a}_l^{(t)}$  denote

	Weight	Lateral	Column	Full
Weight	$\theta_l^{(t',t)}$	$\theta^{(t',t)} = \text{vec}(\theta_l^{(t',t)})_{l \in [L]}$	$\theta^{(t)} = \text{vec}(\theta^{(t',t)})_{t' \in [t]}$	$\theta^{(\leq t)} = \text{vec}(\theta^{(t')})_{t' \in [t]}$
Prior	$\pi_l^{(t',t)}$	$\pi^{(t',t)} = \prod_{l=1}^L \pi_l^{(t',t)}$	$\pi^{(t)} = \prod_{t'=1}^t \pi^{(t',t)}$	$\pi^{(\leq t)} = \prod_{t'=1}^t \pi^{(t')}$
Posterior	$\rho_l^{(t',t)}$	$\rho^{(t',t)} = \prod_{l=1}^L \rho_l^{(t',t)}$	$\rho^{(t)} = \prod_{t'=1}^t \rho^{(t',t)}$	$\rho^{(\leq t)} = \prod_{t'=1}^t \rho^{(t')}$
Fisher matrix	$F_l^{(t',t)}$	$F^{(t',t)} = \text{diag}(F_l^{(t',t)})_{l \in [L]}$	$F^{(t)} = \text{diag}(F^{(t',t)})_{t' \in [t]}$	$F^{(\leq t)} = \text{diag}(F^{(t')})_{t' \in [t]}$

**Table A.1.:** The notation for BPNN. Here,  $1 \leq t' \leq t$  and  $l \in [L]$  for  $t = t'$  and  $l \in \mathcal{L}$  for  $t' < t$ . For task 0, we write (0) instead of (0,0) as no inter-column weights are used.

the activations and pre-activations at layer  $l$  from column  $t$ , then we compute the lateral connection as

$$\mathbf{s}_l^{(t)} = \frac{1}{t} \sum_{t'=1}^t \phi_l^{(t',t)}(\mathbf{a}_{l-1}^{(t')}).$$

Here, we use a superscript  $(t', t)$  to denote the lateral connection from column  $t'$  to  $t$ . We also use this notation for the main column with  $t = t'$ , for the weights, the prior and posterior over the weights, and the Fisher matrix as summarized in section A.1.

We implement PNNs for arbitrary network architectures by storing intermediate activations on the one hand and on the other hand by introducing aggregation layers that combine the stored activations at the layer level. The first part is implemented by creating forward hooks for the lateral connection layers  $l \in \mathcal{L}$  that store the activations of the previous layer  $\mathbf{a}_{l-1}^{(t')}$  during the forward pass in the base network. To combine the activation, the aggregation layers apply the lateral layers to the stored activations of the previous columns,  $\phi_l^{(t',t)}(\mathbf{a}_{l-1}^{(t')})$ , and compute the mean over the resulting pre-activations. The aggregation layer is incorporated into the base network by replacing each layer  $l \in \mathcal{L}$  with the composition of the layer followed by the aggregation layer. With this architectural change, PNNs can be applied to complex network architectures.

**Training of BPNNs.** In Bayesian Progressive Neural Networks (BPNNs), we combine this architecture with Bayesian neural networks and, in particular, LA with our learned prior. Therefore, we start by learning a priori about the dataset  $\mathcal{D}_0$ . This prior is then used as the prior for the main columns excluding the lateral connections. As a priori for the lateral connections, we use the posterior of the corresponding layer of the originating column, as shown in figure 3.1. Unlike the learned prior, this layer was trained to produce reasonable features given the features from the previous layer of the same column, so we use it here instead. After training the prior, the training for each column is similar to LA with the learned prior and curvature scaling. That is, we first optimize the parameters of the column, including all incoming lateral connections, using either MAP estimation or Frequentist projection. During training, we already sample the weights from the previous columns to make the new column robust to small changes in the activations from the lateral connections. After finding the optimal network parameters for a column, we compute the Fisher matrix. We keep the previous weights and distributions fixed, so the Fisher matrix is computed only for the new parameters for the current task. Finally, the curvature is scaled using a PAC-Bayes objective as explained in section 3.2.3. The complete training procedure is also shown in algorithm 9.

**Computational complexity.** For each task and additionally for the prior task, the computational complexity boils down to finding the optimal parameters, computing the Fisher matrix, and scaling the curvature. Parameter optimization is equivalent to finding

```

Input : network architecture  $f$ , datasets  $(\mathcal{D}_t)_{t=0}^T$ ,
         lateral connections  $\mathcal{L}$ , weight decay for task  $\mathfrak{T}_0$ .
Output: The posterior distribution for BPNNs  $\rho^{(\leq T)} = \mathcal{N}(\hat{\boldsymbol{\theta}}^{(\leq T)}, (\tilde{\mathbf{F}}^{(\leq T)})^{-1})$ 

begin
  /* Initialization */
   $\hat{\boldsymbol{\theta}}^{(0)} \leftarrow$  optimal parameters for  $f$  by MAP estimation or a PAC-Bayes objective (section
  A.3.1) using  $\mathcal{D}_0$ .
   $\mathbf{F}^{(0)} \leftarrow$  compute the Fisher using  $\mathcal{D}_0$  ; // Simultaneously compute the negative data
  log-likelihood for  $f_{\hat{\boldsymbol{\theta}}^{(0)}}$ 
   $\alpha^{(0)}, \beta^{(0)}, \tau^{(0)} \leftarrow \arg \min_{\alpha, \beta, \tau} g(\alpha, \beta, \tau)$  ; // here,  $g$  is  $ma$  (equation 3.9) or  $ca$ 
  (equation 3.10)
   $\tilde{\mathbf{F}}_l^{(0)} \leftarrow \frac{1}{\tau_l^{(0)}} (\beta_l^{(0)} \mathbf{F}_l^{(0)} + \alpha_1^{(0)} \gamma \mathbf{I})$  for  $l \in [L]$ 

  /* Main Loop */
  for  $t = 1$  to  $T$  do
    Add a new column and new lateral connections
     $\hat{\boldsymbol{\theta}}^{(t)} \leftarrow$  optimize the new parameters by MAP estimation or a PAC-Bayes objective using
     $\mathcal{D}_t$ 
     $\mathbf{F}^{(t)} \leftarrow$  compute the Fisher using  $\mathcal{D}_t$ 
     $\alpha^{(t)}, \beta^{(t)}, \tau^{(t)} \leftarrow \arg \min_{\alpha, \beta, \tau} g(\alpha, \beta, \tau)$ 
     $\tilde{\mathbf{F}}_l^{(t,t)} \leftarrow \frac{1}{\tau_l^{(t,t)}} (\beta_l^{(t,t)} \mathbf{F}_l^{(t,t)} + \alpha_1^{(t,t)} \tilde{\mathbf{F}}_l^{(0)})$  for  $l \in [L]$ 
     $\tilde{\mathbf{F}}_l^{(i,t)} \leftarrow \frac{1}{\tau_l^{(i,t)}} (\beta_l^{(i,t)} \mathbf{F}_l^{(i,t)} + \alpha_1^{(i,t)} \tilde{\mathbf{F}}_l^{(i,i)})$  for  $i \in [t-1], l \in \mathcal{L}$ 
     $\rho_l^{(i,t)} \leftarrow \mathcal{N}(\hat{\boldsymbol{\theta}}^{(t)}, (\tilde{\mathbf{F}}_l^{(i,t)})^{-1})$  for all available  $i, t, l$ 
  end
end

```

**Algorithm 9:** Training of Bayesian Progressive Neural Networks

the MAP estimate. This can be solved efficiently by computing the negative log-likelihood over mini-batches using common variants of stochastic gradient descent (Kingma & Ba, 2015). In addition, the quadratic term induced by the prior must be computed in each update step. For a diagonal  $\tilde{\mathbf{F}}_l$ , it can be computed with two element-wise multiplications of vectors of size  $n_l$  with  $(\hat{\boldsymbol{\theta}}_l - \tilde{\boldsymbol{\theta}}_l)^T \tilde{\mathbf{F}}_l (\hat{\boldsymbol{\theta}}_l - \tilde{\boldsymbol{\theta}}_l) = (\hat{\boldsymbol{\theta}}_l - \tilde{\boldsymbol{\theta}}_l) \odot \text{diag}(\tilde{\mathbf{F}}_l) \odot (\hat{\boldsymbol{\theta}}_l - \tilde{\boldsymbol{\theta}}_l)$ , where  $\text{diag}(\tilde{\mathbf{F}}_l)$  is the vector containing the diagonal elements of  $\tilde{\mathbf{F}}_l$ . The computations for Kronecker-factored matrices involve two matrix-matrix products with the Kronecker factors and one element-wise product:

$$\begin{aligned}
 (\hat{\boldsymbol{\theta}}_l - \tilde{\boldsymbol{\theta}}_l)^T \tilde{\mathbf{F}}_l (\hat{\boldsymbol{\theta}}_l - \tilde{\boldsymbol{\theta}}_l) &= (\hat{\boldsymbol{\theta}}_l - \tilde{\boldsymbol{\theta}}_l)^T (\mathbf{A} \otimes \mathbf{G}) (\hat{\mathbf{W}}^l - \tilde{\mathbf{W}}^l) \\
 &= (\hat{\boldsymbol{\theta}}_l - \tilde{\boldsymbol{\theta}}_l) \odot \text{vec}(\mathbf{A} (\hat{\mathbf{W}}^l - \tilde{\mathbf{W}}^l) \mathbf{G}),
 \end{aligned}$$

where  $\hat{\mathbf{W}}^l$  and  $\tilde{\mathbf{W}}^l$  are the weight matrices, which correspond to  $\hat{\boldsymbol{\theta}}_l$  and  $\tilde{\boldsymbol{\theta}}_l$  respectively. Hence, this term can be computed efficiently, so that the parameter optimization has a complexity similar to standard MAP training. For common approximations of the Fisher matrix, such as the KFAC Martens (2014) and KFOC Schnaus et al. (2021), the computation of the Fisher matrix corresponds to an additional epoch over the training data (Martens & Grosse, 2015; Grosse & Martens, 2016), although an update step is usually slightly slower than updating the weights. Curvature scaling is an optimization problem with a total of  $3L$  parameters, where  $L$  is the number of layers in the network when all three scales are optimized. Thus, it is usually a much lower-dimensional optimization than finding the weights. In addition, the computation of the negative log-likelihood can be incorporated into

the computation of the Fisher matrix. Therefore, no additional passes through the data are needed. Overall, BPNNs have a similar computational complexity during training as PNNs, the main overhead being the training of an additional task. However, this overhead can be reduced by using a pre-trained model which then leads to the minor overhead of about one additional epoch to compute the Fisher matrix for each task. During inference, BPNNs use multiple network samples, and thus have a computational complexity comparable to multiple forward passes in PNN.

We also comment on the complexity of existing baseline methods. The deep ensemble corresponds to the training ensembles of deep learning models with different initializations and is known to be more expensive than the Kronecker-factored Laplace approximation (Daxberger et al., 2021a). Full LA involves the Hessian, which requires the memory complexity of storing matrices and inverting these matrices, which is cubic in cost. Therefore, full LA is known not to scale, and previous research introduced several approximations such as layer-wise Kronecker-factorization. Of course, learning the prior incurs more cost than relying on an isotropic Gaussian prior.

### A.2.2. Complexity of the Power Method for Sums of Kronecker Products

For all posteriors in BPNN, the final covariance matrix is a sum of Kronecker products. Hence, in general, we are interested in approximating

$$\hat{\mathbf{A}}, \hat{\mathbf{G}} \in \arg \min_{\substack{\mathbf{A} \in \mathbb{R}^{M \times M} \\ \mathbf{G} \in \mathbb{R}^{N \times N}}} \left\| \sum_{k=1}^K \mathbf{A}^k \otimes \mathbf{G}^k - \mathbf{A} \otimes \mathbf{G} \right\|_F, \quad (\text{A.5})$$

for  $M, N, K \in \mathbb{N}$ ,  $\mathbf{A}^k \in \mathbb{R}^{M \times M}$  and  $\mathbf{G}^k \in \mathbb{R}^{N \times N}$  for  $k \in [K]$ . Lemma 3.2.1 shows that this problem is equivalent to a rank-one approximation. Therefore, we can use the power method to solve the problem. However, each step of the plain power method consists of a matrix multiplication with an  $M^2 \times N^2$ -sized matrix. The complexity can be reduced by utilizing that the matrix is a sum of a few rank-one matrices. This is shown in figure 1. With this, the convergence properties of the power method are achieved with a computational complexity of  $\mathcal{O}(n^{\max} K(N^2 + M^2))$ . Also, only  $\mathcal{O}(K(N^2 + M^2))$  memory is needed. Here, we assume that a matrix multiplication  $\mathbf{AB}$  for  $\mathbf{A} \in \mathbb{R}^{m \times n}$  and  $\mathbf{B} \in \mathbb{R}^{n \times k}$  has the complexity  $\mathcal{O}(mnk)$  while a Hadamard product  $\mathbf{A} \odot \mathbf{C}$  for  $\mathbf{C} \in \mathbb{R}^{m \times n}$  can be computed in  $\mathcal{O}(mn)$ .

## A.3. Derivations and Proofs

This section contains proofs for the theory presented. Detailed remarks and follow-up derivations are further provided.

### A.3.1. Derivation of the PAC-Bayes Objectives

In this section, we provide more information on the PAC-Bayes objectives of equation 3.9 and 3.10. In particular, we first derive the upper bound and its approximation shown in equation 3.8 together with computing the KL-divergence in dependence on the curvature scales. This is then used to derive the PAC-Bayes objectives. Next, we show how these objectives can be adapted for network parameter optimization in the Frequentist projection. Finally, we present the extension of the objectives to a continual learning setup, in particular, the proposed BPNNs.

**Upper bound of the expected empirical error.** Here, we show the first equation 3.8:

$$\mathbb{E}_{\boldsymbol{\theta} \sim \rho} \left[ \frac{1}{N} \sum_{i=1}^N \mathbf{1}[\arg \max_{\mathbf{y}'} p(\mathbf{y}' | \mathbf{x}_i, f_{\boldsymbol{\theta}}) \neq \mathbf{y}_i] \right] \leq \mathbb{E}_{\boldsymbol{\theta} \sim \rho} \left[ \frac{1}{N} \sum_{i=1}^N -\frac{\ln p(\mathbf{y}_i | \mathbf{x}_i, f_{\boldsymbol{\theta}})}{\ln 2} \right]. \quad (\text{A.6})$$

For this, as a first step, we present Lemma A.3.1:

**Lemma A.3.1.** *Let  $f : \mathcal{X} \rightarrow \Omega_L$ ,  $(\mathbf{x}, \mathbf{y}) \in \mathcal{X} \times \mathcal{Y}$  and*

$$\text{er}(f, \mathbf{x}, \mathbf{y}) = \mathbf{1}[\arg \max_{\mathbf{y}' \in \mathcal{Y}} p(\mathbf{y}' | \mathbf{x}, f) \neq \mathbf{y}],$$

*then*

$$\text{er}(f, \mathbf{x}, \mathbf{y}) \leq -\frac{\ln p(\mathbf{y} | \mathbf{x}, f)}{\ln 2}.$$

*Proof.* The proof examines the case of a correct prediction first and of a wrong prediction second. For the correct prediction, the error is zero and the bound reduces to the negative log-likelihood being larger than zero. In the case of a wrong prediction, we use that there is a class with a higher probability and that the correct and the most probable class have together a probability that can be bounded by 1 from above.

First, consider the case of a correct classification, which means that  $\mathbf{y}$  has the largest probability:  $\arg \max_{\mathbf{y}' \in \mathcal{Y}} p(\mathbf{y}' | \mathbf{x}, f) = \mathbf{y}$ . Then,  $\text{er}(f, \mathbf{x}, \mathbf{y}) = 0$ . As  $p(\mathbf{y} | \mathbf{x}, f)$  is a discrete probability,  $p(\mathbf{y} | \mathbf{x}, f) \leq 1$  and hence, by taking the negative logarithm on both sides,  $-\frac{\ln p(\mathbf{y} | \mathbf{x}, f)}{\ln 2} \geq 0 = \text{er}(f, \mathbf{x}, \mathbf{y})$ .

Now, let  $\arg \max_{\mathbf{y}' \in \mathcal{Y}} p(\mathbf{y}' | \mathbf{x}, f) \neq \mathbf{y}$ . Therefore,  $\text{er}(f, \mathbf{x}, \mathbf{y}) = 1$  and there exists a  $\mathbf{y}' \in \mathcal{Y}$  such that  $p(\mathbf{y}' | \mathbf{x}, f) \geq p(\mathbf{y} | \mathbf{x}, f)$ . Additionally, we have that  $1 \geq p(\mathbf{y}' | \mathbf{x}, f) + p(\mathbf{y} | \mathbf{x}, f) \geq 2p(\mathbf{y} | \mathbf{x}, f)$ . This can be written as  $p(\mathbf{y} | \mathbf{x}, f) \leq \frac{1}{2}$ . Due to the monotony of the logarithm, we can take the natural logarithm on both sides and divide by  $-\ln 2$  to obtain that  $-\frac{\ln p(\mathbf{y} | \mathbf{x}, f)}{\ln 2} \geq 1 = \text{er}(f, \mathbf{x}, \mathbf{y})$ .

Consequently,  $-\frac{\ln p(\mathbf{y} | \mathbf{x}, f)}{\ln 2}$  is an upper bound of  $\text{er}(f, \mathbf{x}, \mathbf{y})$ .  $\square$

*Remark A.3.2.* We can see from the proof that the bound is tighter when the correct class has a high likelihood. Hence, the bound will be best for good-performing models, while it might be loose when the negative data log-likelihood is large.

We can directly apply Lemma A.3.1 to the empirical error of our neural network  $f_{\boldsymbol{\theta}}$  to get

$$\mathbf{1}[\arg \max_{\mathbf{y}'} p(\mathbf{y}' | \mathbf{x}_i, f_{\boldsymbol{\theta}}) \neq \mathbf{y}_i] = \text{er}(f_{\boldsymbol{\theta}}, \mathbf{x}_i, \mathbf{y}_i) \leq -\frac{\ln p(\mathbf{y}_i | \mathbf{x}_i, f_{\boldsymbol{\theta}})}{\ln 2} \quad (\text{A.7})$$

for each element  $i$  in the dataset. Therefore, also the sum and the expectation are larger or equal, leading to equation A.6.

Moreover, we can plug this upper bound into the PAC-Bayes bounds to receive new upper bounds on the expected loss on the true data distribution:

**Corollary A.3.3.** *Let  $\varepsilon > 0$ ,  $N \in \mathbb{N}$ , and  $\rho \ll \pi$  distributions over the weight-space. Then*

$$L_{\text{upper}} + \sqrt{\frac{\mathbb{KL}(\rho \| \pi) + \ln \frac{2\sqrt{N}}{\varepsilon}}{2N}} \quad \text{and} \quad \inf_{c>0} \frac{1 - \exp(-cL_{\text{upper}} - \frac{\mathbb{KL}(\rho \| \pi) - \ln \varepsilon}{N})}{1 - \exp(-c)}$$

*are upper bounds of the expected error  $\mathbb{E}_{\boldsymbol{\theta} \sim \rho}[\mathcal{L}_P^l(f_{\boldsymbol{\theta}})]$  with probability larger or equal to  $1 - \varepsilon$ .*

*Proof.* Both bounds come from using the upper bound instead of the expected empirical error in the McAllester (Guedj, 2019) and Catoni (Catoni, 2007) bounds. Since both bounds are monotone with respect to the expected empirical error, we get a new bound for each of them.  $\square$

**Approximation of the Expected Empirical Error.** Next, we address the approximation of the upper bound from equation 3.8:

$$\mathbb{E}_{\boldsymbol{\theta} \sim \rho} \left[ \frac{1}{N} \sum_{i=1}^N -\frac{\ln p(\mathbf{y}_i | \mathbf{x}_i, f_{\boldsymbol{\theta}})}{\ln 2} \right] \approx \frac{-\ln p(\mathcal{D} | f_{\hat{\boldsymbol{\theta}}}) + \frac{1}{2} \sum_{l \in [L]} \tau_l \operatorname{tr}(\mathbf{F}_l (\beta_l \mathbf{F}_l + \alpha_l \tilde{\mathbf{F}}_l)^{-1})}{N \ln 2}. \quad (\text{A.8})$$

First, we observe that the upper bound in equation A.6 is the scaled negative data log-likelihood:

$$\mathbb{E}_{\boldsymbol{\theta} \sim \rho} \left[ \frac{1}{N} \sum_{i=1}^N -\frac{\ln p(\mathbf{y}_i | \mathbf{x}_i, f_{\boldsymbol{\theta}})}{\ln 2} \right] = \frac{1}{N \ln 2} \mathbb{E}_{\boldsymbol{\theta} \sim \rho} [-\ln p(\mathcal{D} | f_{\boldsymbol{\theta}})].$$

For the approximation, we use the idea from LA to use the second-order Taylor polynomial around the optimal parameters  $\hat{\boldsymbol{\theta}}$  and replace the Hessian with the Fisher matrix:

$$\begin{aligned} & \frac{1}{N \ln 2} \mathbb{E}_{\boldsymbol{\theta} \sim \rho} [-\ln p(\mathcal{D} | f_{\boldsymbol{\theta}})] \\ & \approx \frac{1}{N \ln 2} \mathbb{E}_{\boldsymbol{\theta} \sim \rho} \left[ -\ln p(\mathcal{D} | f_{\hat{\boldsymbol{\theta}}}) - \mathcal{D} \boldsymbol{\theta}^T (\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}) + \frac{1}{2} (\boldsymbol{\theta} - \hat{\boldsymbol{\theta}})^T \mathbf{F} (\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}) \right]. \end{aligned}$$

In contrast to LA, we do not use the Taylor approximation on the posterior but on the likelihood. Therefore, the quadratic term only includes the Fisher matrix and not the precision of the prior. In the next step, we can move the expectation in because  $\ln p(\mathcal{D} | f_{\hat{\boldsymbol{\theta}}})$  is independent of  $\boldsymbol{\theta}$  and  $\mathbb{E}_{\boldsymbol{\theta} \sim \rho} [\mathcal{D} \boldsymbol{\theta}^T (\boldsymbol{\theta} - \hat{\boldsymbol{\theta}})] = \mathcal{D} \boldsymbol{\theta}^T (\mathbb{E}_{\boldsymbol{\theta} \sim \rho} [\boldsymbol{\theta}] - \hat{\boldsymbol{\theta}}) = \mathcal{D} \boldsymbol{\theta}^T (\hat{\boldsymbol{\theta}} - \hat{\boldsymbol{\theta}}) = 0$ :

$$\begin{aligned} & \frac{\mathbb{E}_{\boldsymbol{\theta} \sim \rho} [-\ln p(\mathcal{D} | f_{\hat{\boldsymbol{\theta}}}) - \mathcal{D} \boldsymbol{\theta}^T (\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}) + \frac{1}{2} (\boldsymbol{\theta} - \hat{\boldsymbol{\theta}})^T \mathbf{F} (\boldsymbol{\theta} - \hat{\boldsymbol{\theta}})]}{N \ln 2} \\ & = \frac{-\ln p(\mathcal{D} | f_{\hat{\boldsymbol{\theta}}}) + \frac{1}{2} \mathbb{E}_{\boldsymbol{\theta} \sim \rho} [(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}})^T \mathbf{F} (\boldsymbol{\theta} - \hat{\boldsymbol{\theta}})]}{N \ln 2} \end{aligned}$$

Finally, we can use that the posterior is a normal distribution  $\rho = \mathcal{N}(\hat{\boldsymbol{\theta}}, \hat{\mathbf{F}}^{-1})$ , where  $\hat{\mathbf{F}}$  is a block-diagonal matrix, where each block is given by  $\hat{\mathbf{F}}_l = \frac{1}{\tau_l} (\beta_l \mathbf{F}_l + \alpha_l \tilde{\mathbf{F}}_l)$ . Therefore, we can reformulate the expectation of the quadratic term as a trace.

$$\begin{aligned} & \frac{-\ln p(\mathcal{D} | f_{\hat{\boldsymbol{\theta}}}) + \frac{1}{2} \mathbb{E}_{\boldsymbol{\theta} \sim \rho} [(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}})^T \mathbf{F} (\boldsymbol{\theta} - \hat{\boldsymbol{\theta}})]}{N \ln 2} \\ & = \frac{-\ln p(\mathcal{D} | f_{\hat{\boldsymbol{\theta}}}) + \frac{1}{2} \sum_{l \in [L]} \tau_l \operatorname{tr}(\mathbf{F}_l (\beta_l \mathbf{F}_l + \alpha_l \tilde{\mathbf{F}}_l)^{-1})}{N \ln 2} =: \operatorname{aer}(\alpha, \beta, \tau). \end{aligned}$$

This approximation depends only on quantities that were already computed during the LA, such as the Fisher matrix and the optimal parameters, or that can be computed without extra effort during the computation of the LA, such as the negative data log-likelihood for the optimal parameters.

**KL-divergence.** The PAC-Bayes bounds are dependent not only on the expected empirical risk but also on the KL-divergence between the prior and posterior. Since we use LA for both, this boils down to the KL-divergence between two multivariate normal



distributions, which can be computed in closed form. For the prior  $\rho = \mathcal{N}(\tilde{\boldsymbol{\theta}}, \tilde{\mathbf{F}}^{-1})$  and posterior  $\mathcal{N}(\hat{\boldsymbol{\theta}}, \hat{\mathbf{F}}^{-1})$  defined as above, the KL-divergence can be computed as

$$\begin{aligned}
\mathbb{KL}(\rho \parallel \pi) &= \mathbb{KL}(\mathcal{N}(\hat{\boldsymbol{\theta}}, \hat{\mathbf{F}}^{-1}) \parallel \mathcal{N}(\tilde{\boldsymbol{\theta}}, \tilde{\mathbf{F}}^{-1})) \\
&= \frac{1}{2} \left( \text{tr}(\tilde{\mathbf{F}} \hat{\mathbf{F}}^{-1}) - n - \ln \frac{\det \tilde{\mathbf{F}}}{\det \hat{\mathbf{F}}} + (\hat{\boldsymbol{\theta}} - \tilde{\boldsymbol{\theta}})^T \tilde{\mathbf{F}} (\hat{\boldsymbol{\theta}} - \tilde{\boldsymbol{\theta}}) \right) \\
&= \frac{1}{2} \left( \sum_{l \in [L]} \text{tr}(\tilde{\mathbf{F}}_l \tau_l (\beta_l \mathbf{F}_l + \alpha_l \tilde{\mathbf{F}}_l)^{-1}) - n_l - \ln \frac{\det \tilde{\mathbf{F}}_l}{\det(\frac{1}{\tau_l} (\beta_l \mathbf{F}_l + \alpha_l \tilde{\mathbf{F}}_l))} + (\hat{\boldsymbol{\theta}}_l - \tilde{\boldsymbol{\theta}}_l)^T \tilde{\mathbf{F}}_l (\hat{\boldsymbol{\theta}}_l - \tilde{\boldsymbol{\theta}}_l) \right) \\
&= \frac{1}{2} \sum_{l \in [L]} \tau_l \text{tr}(\tilde{\mathbf{F}}_l (\beta_l \mathbf{F}_l + \alpha_l \tilde{\mathbf{F}}_l)^{-1}) \\
&\quad - \frac{1}{2} n_l (1 + \ln \tau_l) - \ln \frac{\det \tilde{\mathbf{F}}_l}{\det(\beta_l \mathbf{F}_l + \frac{1}{2} \alpha_l \tilde{\mathbf{F}}_l)} + \frac{1}{2} (\hat{\boldsymbol{\theta}}_l - \tilde{\boldsymbol{\theta}}_l)^T \tilde{\mathbf{F}}_l (\hat{\boldsymbol{\theta}}_l - \tilde{\boldsymbol{\theta}}_l) \\
&=: \text{kl}(\alpha, \beta, \tau). \tag{A.9}
\end{aligned}$$

Similar to the approximation of the expected empirical error, all relevant quantities to compute the KL-divergence are already given after the LA.

**PAC-Bayes objectives.** Combining the approximation of the expected empirical error  $\text{aer}(\alpha, \beta, \tau)$  and the KL-divergence  $\text{kl}(\alpha, \beta, \tau)$ , we get approximations of the McAllester bound (Guedj, 2019):

$$ma(\alpha, \beta, \tau) = \text{aer}(\alpha, \beta, \tau) + \sqrt{\frac{\text{kl}(\alpha, \beta, \tau) + \ln \frac{2\sqrt{N}}{\varepsilon}}{2N}},$$

and Catoni bound (Catoni, 2007):

$$ca(\alpha, \beta, \tau) = \inf_{c>0} \frac{1 - \exp(-c \text{aer}(\alpha, \beta, \tau) - \frac{\text{kl}(\alpha, \beta, \tau) - \ln \varepsilon}{N})}{1 - \exp(-c)}.$$

These objectives can be evaluated and minimized without going through any data samples.

### Frequentist Projection

In addition to the curvature scaling, we also propose Frequentist projection, which also uses approximations of the bounds to optimize the network parameters. This has the goal of further minimizing the PAC-Bayesian bounds also with the choice of the parameters and not only with the curvature scaling. Nonetheless, as the Fisher matrix is only available after the weights are found, we neglect the terms that depend on the curvature. For the McAllester Bound, this results in the optimization problem

$$\min_{\boldsymbol{\theta}} -\frac{\ln p(\mathcal{D}|f_{\boldsymbol{\theta}})}{\ln(2)N} + \sqrt{\frac{\frac{1}{2\tau}(\boldsymbol{\theta} - \tilde{\boldsymbol{\theta}})^T \tilde{\mathbf{F}}(\boldsymbol{\theta} - \tilde{\boldsymbol{\theta}}) + \ln(\frac{2N}{\varepsilon})}{2N}}.$$

The Catoni Bound is difficult to optimize with variations of stochastic gradient descent because of the exponential in the objective function. Nonetheless, when the Catoni scale  $c > 0$  is fixed, minimizing the upper bound of the Catoni Bound is equivalent to calculating

$$\min_{\boldsymbol{\theta}} -c \frac{\ln p(\mathcal{D}|f_{\boldsymbol{\theta}})}{\ln(2)N} + \frac{\frac{1}{2\tau}(\boldsymbol{\theta} - \tilde{\boldsymbol{\theta}})^T \tilde{\mathbf{F}}(\boldsymbol{\theta} - \tilde{\boldsymbol{\theta}}) - \ln \varepsilon}{N}.$$

Therefore, we heuristically alternate between optimizing  $\boldsymbol{\theta}$  and  $c$  in practice, where the optimization after the Catoni scale is done using the objective

$$\min_{c>0} \frac{1 - \exp\left(c \frac{\ln p(\mathcal{D}|f_{\boldsymbol{\theta}})}{\ln(2)N} - \frac{\frac{1}{2\tau}(\boldsymbol{\theta} - \tilde{\boldsymbol{\theta}})^T \tilde{\mathbf{F}}(\boldsymbol{\theta} - \tilde{\boldsymbol{\theta}}) - \ln \varepsilon}{N}\right)}{1 - \exp(-c)}.$$

### Extension to BPNNs

For BPNNs, we also have to consider the parameters and posteriors of the previous distributions. Therefore, in this section, we present the adaptations for BPNNs. Since we still use the same approximations as in the transfer learning setup, we need to estimate the upper bound of the expected empirical risk and the KL-divergence as a function of the curvature scales. We reuse the notation from the appendix A.2. Hence, we denote the correspondence to a given column by a superscript. In addition, we denote the precision matrix of the posterior with a tilde, i.e.,  $\tilde{\mathbf{F}}_l^{(i,t)} = \frac{1}{\tau_l}(\beta_l^{(i,t)} \mathbf{F}_l^{(i,t)} + \alpha_l^{(i,t)} \tilde{\mathbf{F}}_l^{(i,i)})$  for a lateral connection in layer  $l$  from column  $i$  to column  $j$ . One can see from  $\tilde{\mathbf{F}}_l^{(i,i)}$  that we use the posterior of the column  $i$  as the prior for this lateral connection.

**Approximation of the expected empirical error.** To compute the approximate upper bound of the expected empirical error, we need to compute the expected empirical error on the training data as in equation A.8. For the proposed continual learning architecture, we can compute the upper bound of the expected empirical error and its approximation with

$$\begin{aligned} & \mathbb{E}_{\boldsymbol{\theta}^{(\leq t)} \sim \rho^{(\leq t)}} \left[ \frac{1}{N_t} \sum_{i=1}^{N_t} \text{er}(f_{\boldsymbol{\theta}^{(\leq t)}}, \mathbf{x}_i^{(t)}, \mathbf{y}_i^{(t)}) \right] \\ & \leq \frac{1}{N_t \ln 2} \mathbb{E}_{\boldsymbol{\theta}^{(\leq t)} \sim \rho^{(\leq t)}} \left[ -\ln p(\mathcal{D}_t | f_{\boldsymbol{\theta}^{(\leq t)}}) \right] \\ & \approx \frac{1}{N_t \ln 2} \left( -\ln p(\mathcal{D}_t | f_{\hat{\boldsymbol{\theta}}^{(\leq t)}}) + \frac{1}{2} \mathbb{E}_{\boldsymbol{\theta}^{(\leq t)} \sim \rho^{(\leq t)}} \left[ (\boldsymbol{\theta}^{(\leq t)} - \hat{\boldsymbol{\theta}}^{(\leq t)})^T \mathbf{F}^{(\leq t)} (\boldsymbol{\theta}^{(\leq t)} - \hat{\boldsymbol{\theta}}^{(\leq t)}) \right] \right), \end{aligned}$$

where  $\mathbf{F}^{(\leq t)}$  denotes the Fisher matrix on task  $t$  for all weights up to column  $t$ . The quadratic term in the expectation can be computed exactly, i.e., we can use  $\rho^{(\leq t)}$  as a normal distribution with mean  $\hat{\boldsymbol{\theta}}^{(\leq t)}$ . Moreover, we can decompose this into:

$$\begin{aligned} & \mathbb{E}_{\boldsymbol{\theta}^{(\leq t)} \sim \rho^{(\leq t)}} \left[ (\boldsymbol{\theta}^{(\leq t)} - \hat{\boldsymbol{\theta}}^{(\leq t)})^T \mathbf{F}^{(\leq t)} (\boldsymbol{\theta}^{(\leq t)} - \hat{\boldsymbol{\theta}}^{(\leq t)}) \right] = \text{tr}(\mathbf{F}^{(\leq t)} (\tilde{\mathbf{F}}^{(\leq t)})^{-1}) \\ & = \sum_{j=1}^{t-1} \sum_{i=1}^j \text{tr}(\mathbf{F}^{(i,j)} (\tilde{\mathbf{F}}^{(i,j)})^{-1}) + \sum_{i=1}^t \text{tr}(\mathbf{F}^{(i,t)} (\tilde{\mathbf{F}}^{(i,t)})^{-1}). \end{aligned}$$

The weights from the previous columns, *i.e.* for  $i \leq j < t$ , are frozen, and their corresponding posterior scales together with their curvature scales are fixed. Therefore, the first term is constant with respect to the new curvature scales. To avoid computing the Fisher matrix with respect to previous network weights, we reuse the fixed Fisher matrix from the

corresponding task of the given column. This simplifies the upper bound:

$$\begin{aligned}
& \sum_{j=1}^{t-1} \sum_{i=1}^j \text{tr}(\mathbf{F}^{(i,j)} (\tilde{\mathbf{F}}^{(i,j)})^{-1}) + \sum_{i=1}^t \text{tr}(\mathbf{F}^{(i,t)} (\tilde{\mathbf{F}}^{(i,t)})^{-1}) \\
& \approx \text{tr}^{(\leq t-1)} + \sum_{i=1}^t \text{tr}(\mathbf{F}^{(i,t)} (\tilde{\mathbf{F}}^{(i,t)})^{-1}), \\
& = \text{tr}^{(\leq t-1)} + \sum_{i=1}^{t-1} \sum_{l \in \mathfrak{L}} \text{tr}(\mathbf{F}_l^{(i,t)} \tau_l(\beta_l^{(i,t)} \mathbf{F}_l^{(i,t)} + \alpha_l^{(i,t)} \tilde{\mathbf{F}}_l^{(i,i)})^{-1}) \\
& \quad + \sum_{l=1}^L \text{tr}(\mathbf{F}_l^{(t,t)} \tau_l(\beta_l^{(t,t)} \mathbf{F}_l^{(t,t)} + \alpha_l^{(t,t)} \tilde{\mathbf{F}}_l^{(0)})^{-1}).
\end{aligned}$$

Here, we introduce the trace for all fixed previous columns and Fisher matrices as  $\text{tr}^{(\leq t-1)}$ . This term is not dependent on the curvature scales. Moreover, the set of lateral connections is  $\mathfrak{L}$ .

**KL-divergence.** With a block-diagonal covariance matrix, we assume that the weights of different layers are independent. Using this, the KL-divergence between the posterior and the prior can be written as

$$\mathbb{KL}(\rho^{(\leq t)} \parallel \pi^{(\leq t)}) = \sum_{j=1}^t \sum_{i=1}^j \mathbb{KL}(\rho^{(i,j)} \parallel \pi^{(i,j)}) \quad (\text{A.10})$$

$$= \sum_{i=1}^t \mathbb{KL}(\rho^{(i,t)} \parallel \pi^{(i,t)}) + \sum_{j=1}^{t-1} \sum_{i=1}^j \mathbb{KL}(\rho^{(i,j)} \parallel \pi^{(i,j)}) \quad (\text{A.11})$$

$$= \sum_{i=1}^t \mathbb{KL}(\rho^{(i,t)} \parallel \pi^{(i,t)}) \quad (\text{A.12})$$

$$= \sum_{l \in [L]} \mathbb{KL}(\rho_l^{(t,t)} \parallel \mathcal{N}(\hat{\boldsymbol{\theta}}_l^{(0)}, (\tilde{\mathbf{F}}_l^{(0)})^{-1})) + \sum_{l \in \mathfrak{L}} \sum_{i=1}^{t-1} \mathbb{KL}(\rho_l^{(i,t)} \parallel \rho_l^{(i,i)}),$$

where it was used that the prior is equal to the posterior for all fixed columns in equation A.10. We approximate the posterior with LA and curvature scaling. Therefore, we can compute the KL-divergence again in closed form:

$$\mathbb{KL}(\rho^{(\leq t)} \parallel \pi^{(\leq t)}) = \sum_{l \in [L]} \mathbb{KL}(\mathcal{N}(\hat{\boldsymbol{\theta}}_l^{(t,t)}, \tau_l(\beta_l^{(t,t)} \mathbf{F}_l^{(t,t)} + \alpha_l^{(t,t)} \tilde{\mathbf{F}}_l^{(0)})^{-1}) \parallel \mathcal{N}(\hat{\boldsymbol{\theta}}_l^{(0)}, (\tilde{\mathbf{F}}_l^{(0)})^{-1})) \quad (\text{A.13})$$

$$\mathbb{KL}(\rho^{(\leq t)} \parallel \pi^{(\leq t)}) + \sum_{l \in \mathfrak{L}} \sum_{i=1}^{t-1} \mathbb{KL}(\mathcal{N}(\hat{\boldsymbol{\theta}}_l^{(i,t)}, \tau_l(\beta_l^{(i,t)} \mathbf{F}_l^{(i,t)} + \alpha_l^{(i,t)} \tilde{\mathbf{F}}_l^{(i,i)})^{-1}) \parallel \mathcal{N}(\hat{\boldsymbol{\theta}}_l^{(i,i)}, (\tilde{\mathbf{F}}_l^{(i,i)})^{-1})) \quad (\text{A.14})$$

Altogether, this shows the advantage of BPNN that some related feature embeddings can increase the performance even though only the current column contributes to the KL-divergence.

### A.3.2. Proof of Lemma 3.2.1

**Lemma 3.2.1.** *Let  $M, N, K \in \mathbb{N}$ ,  $\mathbf{A}^k \in \mathbb{R}^{M \times M}$  and  $\mathbf{G}^k \in \mathbb{R}^{N \times N}$  for  $k \in [K]$ . Then*

$$\left\| \sum_{k=1}^K \mathbf{A}^k \otimes \mathbf{G}^k - \mathbf{A} \otimes \mathbf{G} \right\|_F = \left\| \sum_{k=1}^K \text{vec}(\mathbf{A}^k) \text{vec}(\mathbf{G}^k)^T - \text{vec}(\mathbf{A}) \text{vec}(\mathbf{G})^T \right\|_F. \quad (3.5)$$

*Proof.* Let  $i, j \in [MN]$ . Then the  $i, j$ -th entry of the left matrix is

$$\begin{aligned} \left( \sum_{k=1}^K \mathbf{A}^k \otimes \mathbf{G}^k - \mathbf{A} \otimes \mathbf{G} \right)_{i,j} &= \sum_{k=1}^K \mathbf{A}_{i_1, j_1}^k \mathbf{G}_{i_2, j_2}^k - \mathbf{A}_{i_1, j_1} \mathbf{G}_{i_2, j_2} \\ &= \left( \sum_{k=1}^K \text{vec}(\mathbf{A}^k) \text{vec}(\mathbf{G}^k)^T - \text{vec}(\mathbf{A}) \text{vec}(\mathbf{G})^T \right)_{M(i_1-1)+j_1, N(i_2-1)+j_2}, \end{aligned} \quad (A.15)$$

with  $i = N(i_1 - 1) + i_2$ ,  $j = N(j_1 - 1) + j_2$ . Hence, both matrices have the same entries and only the order of the entries is in general different. Therefore, the sum over the squared entries and thus the Frobenius norm is the same:

$$\begin{aligned} \left\| \sum_{k=1}^K \mathbf{A}^k \otimes \mathbf{G}^k - \mathbf{A} \otimes \mathbf{G} \right\|_F^2 &= \sum_{i=1}^{M^2} \sum_{j=1}^{N^2} \left( \sum_{k=1}^K \mathbf{A}^k \otimes \mathbf{G}^k - \mathbf{A} \otimes \mathbf{G} \right)_{i,j}^2 \\ &= \sum_{i_1, j_1=1}^N \sum_{i_2, j_2=1}^M \left( \sum_{k=1}^K \mathbf{A}^k \otimes \mathbf{G}^k - \mathbf{A} \otimes \mathbf{G} \right)_{N(i_1-1)+i_2, N(j_1-1)+j_2}^2 \\ &= \sum_{i_1, j_1=1}^N \sum_{i_2, j_2=1}^M \left( \sum_{k=1}^K \mathbf{A}_{i_1, j_1}^k \otimes \mathbf{G}_{i_2, j_2}^k - \mathbf{A}_{i_1, j_1} \otimes \mathbf{G}_{i_2, j_2} \right)^2 \\ &= \sum_{i_1, j_1=1}^N \sum_{i_2, j_2=1}^M \left( \sum_{k=1}^K \text{vec}(\mathbf{A}^k) \text{vec}(\mathbf{G}^k)^T - \text{vec}(\mathbf{A}) \text{vec}(\mathbf{G})^T \right)_{M(i_1-1)+j_1, N(i_2-1)+j_2}^2 \\ &= \left\| \sum_{k=1}^K \text{vec}(\mathbf{A}^k) \text{vec}(\mathbf{G}^k)^T - \text{vec}(\mathbf{A}) \text{vec}(\mathbf{G})^T \right\|_F^2. \end{aligned} \quad (A.16)$$

□

### A.3.3. Proof of Lemma 3.2.2

**Lemma 3.2.2.** *Let  $\mathbf{L} = \sum_{k=1}^K \text{vec}(\mathbf{A}^k) \text{vec}(\mathbf{G}^k)^T$  and  $\mathbf{L} = \sum_{i=1}^r \sigma_i \mathbf{u}_i \mathbf{v}_i^T$  be its singular value decomposition with  $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_r > 0$  and  $\mathbf{u}_i^T \mathbf{u}_j = \mathbf{v}_i^T \mathbf{v}_j = \mathbf{1}[i = j]$ . Then there is a solution of the equation 3.4 with*

$$\text{vec}(\hat{\mathbf{A}}) = \mathbf{u}_1, \text{vec}(\hat{\mathbf{G}}) = \sigma_1 \mathbf{v}_1. \quad (A.17)$$

*If  $\sigma_1 > \sigma_2$ , the solution is unique up to changing the sign of both factors, and Algorithm 1 converges almost surely to this solution.*

*Proof.* The main idea of the proof is to use Lemma 3.2.1 to identify the problem with a best rank-one approximation. The algorithm then corresponds to the power method that utilizes the Kronecker factorization for a faster and memory-efficient computation of the matrix-vector products in the Kronecker matrix space.

By the Eckart-Young-Mirsky theorem (Eckart & Young, 1936), an optimal rank-one approximation for  $A$  in the Frobenius norm is

$$\sigma_1 \mathbf{u}_1 \mathbf{v}_1^T \in \arg \min_{\hat{\mathbf{L}} \in \mathbb{R}^{M^2 \times N^2} : \text{rank}(\hat{\mathbf{L}})=1} \|\mathbf{L} - \hat{\mathbf{L}}\|_F, \quad (\text{A.18})$$

which is unique up to changing the sign of both factors if  $\sigma_1 > \sigma_2$ .

Therefore, the matrices  $\hat{\mathbf{A}}$  and  $\hat{\mathbf{G}}$  that satisfy equation (A.17) are optimal solutions of equation (3.4). Moreover, the left factor is normalized, e.g.  $\|\hat{\mathbf{A}}\|_F = \|\mathbf{u}_1\|_2 = 1$ .

The equivalence of Algorithm 1 with the power method can be seen by multiplying  $\mathbf{L}\mathbf{L}^T$  with  $\text{vec}(\mathbf{A}^{(n-1)})$  for  $\mathbf{A}^{(n-1)} \in \mathbb{R}^{N^2}$ :

$$\mathbf{L}^T \text{vec}(\mathbf{A}^{(n-1)}) = \sum_{k=1}^K \text{vec}(\mathbf{G}^k) \text{vec}(\mathbf{A}^k)^T \text{vec}(\mathbf{A}^{(n-1)}) \quad (\text{A.19})$$

$$= \sum_{k=1}^K \langle \mathbf{A}^k, \mathbf{A}^{(n-1)} \rangle_F \text{vec}(\mathbf{G}^k) \quad (\text{A.20})$$

$$= \text{vec}(\bar{\mathbf{G}}^{(n)}), \quad (\text{A.21})$$

and

$$\mathbf{L}\mathbf{L}^T \text{vec}(\mathbf{A}^{(n-1)}) = \sum_{k=1}^K \text{vec}(\mathbf{A}^k) \text{vec}(\mathbf{G}^k)^T \text{vec}(\bar{\mathbf{G}}^{(n)}) \quad (\text{A.22})$$

$$= \sum_{k=1}^K \langle \mathbf{G}^k, \bar{\mathbf{G}}^{(n)} \rangle_F \text{vec}(\mathbf{A}^k) \quad (\text{A.23})$$

$$= \|\bar{\mathbf{G}}^{(n)}\|_F \sum_{k=1}^K \langle \mathbf{G}^k, \bar{\mathbf{G}}^{(n)} \rangle_F \text{vec}(\mathbf{A}^k) \quad (\text{A.24})$$

$$= \|\bar{\mathbf{G}}^{(n)}\|_F \text{vec}(\bar{\mathbf{A}}^{(n)}). \quad (\text{A.25})$$

Hence, we can compute the same iterations as the standard power method, like Algorithm 1:

$$\frac{\mathbf{L}\mathbf{L}^T \text{vec}(\mathbf{A}^{(n-1)})}{\|\mathbf{L}\mathbf{L}^T \text{vec}(\mathbf{A}^{(n-1)})\|_2} = \frac{\|\bar{\mathbf{G}}^{(n)}\|_F \text{vec}(\bar{\mathbf{A}}^{(n)})}{\|\bar{\mathbf{G}}^{(n)}\|_F \|\bar{\mathbf{A}}^{(n)}\|_F} = \frac{\text{vec}(\bar{\mathbf{A}}^{(n)})}{\|\bar{\mathbf{A}}^{(n)}\|_F} = \text{vec}(\mathbf{A}^{(n)}). \quad (\text{A.26})$$

The final right factor then corresponds to  $\mathbf{L}^T \text{vec}(\mathbf{A}^{(n)}) \approx \sigma_1 \mathbf{v}_1$ .

For  $\sigma_1 > \sigma_2$ , the convergence properties are inherited from the power method, e.g., see the work of Bindel (2016).  $\square$

*Remark A.3.4.* Even in the case when the first singular value is not (much) larger than the other singular values and no convergence is achieved, the resulting matrices of Algorithm 1 are, with high probability, in the span of the singular vectors corresponding to the set of large singular values (Blum et al., 2020). Hence, in this case, the approximation will still converge to good Kronecker factors with high probability.

## A.4. Some Thoughts on the Regression

In this section, we share potential extensions of the given PAC-Bayes framework in LA, to regression problems. PAC-Bayes traditionally assumed bounded losses as in classification

problems, while in recent years the theory has also been extended to unbounded loss functions, as in regression problems. Although we see this extension as beyond the scope of this work, we comment on some thoughts on the regression case. For this, let us start with a general bound from [Alquier et al. \(2016\)](#):

**Definition A.4.1** ([Alquier et al. \(2016\)](#)). Given a distribution  $\mathcal{D}$  over  $\mathcal{X} \times \mathcal{Y}$ , a hypothesis set  $\mathcal{G}$ , a loss function  $l : \mathcal{G} \times \mathcal{X} \times \mathcal{Y}$ , a prior distribution  $\pi$  over  $\mathcal{G}$ , a  $\delta \in (0, 1]$  and a real number  $\gamma > 0$ , with probability at least  $1 - \delta$  over the choice of  $(X, Y) \sim \mathcal{D}^n$ , we have

$$\forall \rho \text{ on } \mathcal{G} : \quad \mathbb{E}_{g \sim \rho} \mathcal{L}_{\mathcal{D}}^l(g) \leq \mathbb{E}_{g \sim \rho} \hat{\mathcal{L}}_{\mathbf{X}, \mathbf{Y}}^l(g) + \frac{1}{\lambda} \left[ \text{KL}(\rho \| \pi) + \ln \frac{1}{\delta} + \Psi_{l, \pi, \mathcal{D}}(\lambda, n) \right], \quad (\text{A.27})$$

$$\text{where } \Psi_{l, \pi, \mathcal{D}}(\lambda, n) = \ln \mathbb{E}_{g \sim \pi} \mathbb{E}_{X', Y' \sim \mathcal{D}^n} \exp \left[ \lambda (\mathcal{L}_{\mathcal{D}}^l(g) - \hat{\mathcal{L}}_{X', Y'}^l(g)) \right]. \quad (\text{A.28})$$

A special case for the unbounded loss functions has been examined in [Germain et al. \(2016\)](#). The results showed that, for regression problems, the Alquier bound holds for certain classes of loss functions. One of them is the so-called sub-gamma losses. Sub-gamma losses, given variance factor  $s^2$  and scale parameter  $c$ , have a variable  $\psi_V(\lambda) := \ln \mathbb{E} e^{\lambda V}$  with  $V = \mathcal{L}_{\mathcal{D}}^l(g) - l(g, x, y)$ :

$$\psi_V(\lambda) \leq \frac{s^2}{c^2} (-\ln(1 - \lambda c)) \leq \frac{\lambda^2 s^2}{2(1 - c\lambda)}, \quad \lambda \in (0, \frac{1}{c}). \quad (\text{A.29})$$

For these types of losses, the PAC-Bayes bound can be obtained as shown in the theorem below.

**Definition A.4.2** ([Germain et al. \(2016\)](#)). Given  $D, \mathcal{G}, l, \pi$  and  $\delta$  defined in the statement of the above theorem (definition A.4.1), if the loss is sub-gamma with variance factor  $s^2$  and scale  $c < 1$  we have, with probability at least  $1 - \delta$  over  $(\mathbf{X}, \mathbf{Y}) \sim \mathcal{D}^n$ ,

$$\forall \rho \text{ on } \mathcal{G} : \quad \mathbb{E}_{g \sim \rho} \mathcal{L}_{\mathcal{D}}^l(g) \leq \mathbb{E}_{g \sim \rho} \hat{\mathcal{L}}_{\mathbf{X}, \mathbf{Y}}^l(g) + \frac{1}{N} \left[ \text{KL}(\rho \| \pi) + \ln \frac{1}{\delta} \right] + \frac{s^2}{2(1 - c)}. \quad (\text{A.30})$$

One concrete example is the case for Bayesian linear regression. Here, our model is:  $f_{\boldsymbol{\theta}}(\mathbf{x}) = \boldsymbol{\theta} \phi(\mathbf{x})$  with the modeling choices of:  $\boldsymbol{\theta} \sim \mathcal{N}(\mathbf{0}, \sigma_{\pi}^2 \mathbf{I})$  and  $\phi(\mathbf{x}) \sim \mathcal{N}(\mathbf{0}, \sigma_x^2 \mathbf{I})$ . Also, we add white noise:  $\mathbf{y} = \hat{\boldsymbol{\theta}} \mathbf{x} + \epsilon$  where  $\epsilon \sim \mathcal{N}(\mathbf{0}, \sigma_{\epsilon}^2)$ , resulting in  $p(\mathbf{y} | \mathbf{x}) \sim \mathcal{N}(\boldsymbol{\Phi}(\mathbf{x}) \cdot \hat{\boldsymbol{\theta}}, \sigma_{\epsilon}^2)$ . As in [Bishop & Nasrabadi \(2006\)](#),  $p(\boldsymbol{\theta} | \mathcal{D}, \sigma, \sigma_{\pi}) = \mathcal{N}(\boldsymbol{\theta} | \hat{\boldsymbol{\theta}}, \boldsymbol{\Sigma})$  where  $\boldsymbol{\Sigma}^{-1} := \frac{1}{\sigma^2} \boldsymbol{\Phi}^T \boldsymbol{\Phi} + \frac{1}{\sigma_{\pi}^2} \mathbf{I}$ . Under these settings and using negative log likelihood, [Germain et al. \(2016\)](#) shows:

$$s^2 \geq \frac{1}{\lambda \sigma^2} \left[ \sigma_x^2 (\sigma_{\pi}^2 d + \|\hat{\boldsymbol{\theta}}\|^2) + \sigma_{\epsilon}^2 (1 - \lambda c) \right] \quad \text{and} \quad c \geq \frac{1}{\sigma^2} (\sigma_x^2 \sigma_{\pi}^2). \quad (\text{A.31})$$

Moreover, under Gaussian assumptions, the Alquier bound is closed, e.g.,  $N \mathbb{E}_{\boldsymbol{\theta} \sim \rho} \hat{\mathcal{L}}_{\mathbf{X}, \mathbf{Y}}^{l_{\text{nl}}}(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{\theta} \sim \rho} \sum_{i=1}^n -\ln p(\mathbf{y}_i | \mathbf{x}_i, \boldsymbol{\theta}) = N \hat{\mathcal{L}}_{\mathbf{X}, \mathbf{Y}}^{l_{\text{nl}}}(\boldsymbol{\theta}) + \frac{1}{2\sigma^2} \text{tr}(\boldsymbol{\Phi}^T \boldsymbol{\Phi} \boldsymbol{\Sigma})$ , and also the KL-divergence term between two Gaussian distributions. These results are given generalized linear models, while our work is about neural networks. On the other hand, if neural networks can be approximated with linear regression in some sense, we should intuitively be able to exploit the results so far. Hence, we next make the connections between BNNs and linear regression via LA. The linearization scheme has been presented in chapter 2. From chapter 5 we also have the work of [Khan et al. \(2019\)](#).

**Definition A.4.3** ([Khan et al. \(2019\)](#)). The Laplace Approximation of a neural network posterior  $p(\boldsymbol{\theta} | \mathcal{D})$  is equal to the posterior distribution  $p(\boldsymbol{\theta} | \hat{\mathcal{D}})$ .

This means that we may then bring these results together in order to obtain a valid PAC-Bayes bound. Overall, there exists a valid PAC-Bayes bound for Bayesian linear regression models that can be obtained in closed form for tractable optimization. As with LA, a neural network has a linear subspace defined by a Bayesian linear model. One could also analyze LA-based BNNs like (Lee et al., 2021) using the PAC-Bayes bound and consequently the curvature scaling.

## A.5. Implementation Details

Now, we present the implementation details that have been used in our experiments. We plan to officially release the code to the community. While implementation details are provided for different experiments, the general setting is as follows. Our software is based on Pytorch when possible, while only in few-shot learning experiments, we also use TensorFlow and JAX in order to utilize the 'uncertainty baselines' repository. In all our experiments, the Fisher matrix is computed with the K-FOC method (Schnaus et al., 2021), which is a subclass of the KFAC method from section 3.2. Our computing cluster consists of multiple GPUs, from NVIDIA's Pascal to Ampere architecture. Multi-GPU training was not used for the results. The largest GPU memory is 24GB, while CPU RAM of 120GB is available for these large GPUs.

### A.5.1. Implementation Details: Section 3.4.1

The implementation details for section 3.4.1 are as follows. We use 90% of the training data for training and 10% for validation. Furthermore, we use the Adam optimizer with a batch size of 256 and a learning rate of  $5 \cdot 10^{-4}$ . The learning rate is halved if the validation loss does not decrease for five consecutive steps, and training is stopped after 100 epochs or if the loss does not improve for ten consecutive steps. We also optimize the curvature scales using Adam with an exponential learning rate decay with a decay factor of 0.999999 per step. In our experiments, we compare the performance of our learned prior against an isotropic prior with weight decay  $10^{-3}$ ,  $10^{-5}$ , and  $10^{-8}$  either with mean zero or using the pre-trained model as mean. We also use the same weight decay to compute the learned prior. For all Bayesian neural networks, we use 100 samples from the posterior to estimate the predictive distribution. The PAC-Bayes bounds are evaluated with  $\varepsilon = 0.1$ . Hence, the bounds are satisfied with at least 90% probability.

We note that cross-validation would be a reasonable method to use the entire training set. However, this would mean that with  $k$  folds, one would have to retrain the model  $k$  times with each hyperparameter configuration. Even with only one separate validation set, grid search would quickly become infeasible when optimizing three parameters per layer, each with multiple values. This is primarily because one would need to sample multiple weights for each validation sample to estimate the predictive distribution, and then repeat this for the entire validation set for each hyperparameter optimization. Meanwhile, for PAC-Bayes objectives, we reuse some of the quantities that can be estimated during the Laplace approximation (the Fisher matrix, the optimal network parameters, and the log-likelihood of the training data using the optimal parameters). In this way, our objectives can be evaluated without additional forward passes. Therefore, the complexity of hyperparameter optimization is independent of the size of the dataset and the complexity of the model forward pass, and thus in practice, its convergence can be faster than grid search. To illustrate this point, we compared to a grid search over  $\alpha \in \{0.01, 0.1, 0.3, 0.5, 0.8, 0.9, 0.99, 1, 2\}$ ,



$\beta \in \{0.01, 0.1, 0.3, 0.5, 0.8, 0.9, 0.99, 1, 2\}$ , and  $\tau \in \{10^{-i} | i \in [26]\}$  shared over each layer. The models maximizing the validation accuracy were chosen. Other ablations are similar to the setting of figure 2a and we have used five different seeds. Thus, curvature scaling and Frequentist projection refer to our PAC-Bayes-based approaches. The results are shown in the corresponding table.

We also quantitatively evaluated the quality of the approximation on the sums-of-Kronecker-products. For this, we measured the relative error in terms of a Frobenius norm, where we compared the approximates to the true sums-of-Kronecker product with positive definite matrices. The baseline is the approximation adapted by Ritter et al. (2018b,a), which assumes  $(\mathbf{A} \otimes \mathbf{G} + \gamma \mathbf{I})^{-1} \approx (\mathbf{A} + \gamma \mathbf{I})^{-1} \otimes (\mathbf{G} + \gamma \mathbf{I})^{-1}$  (denoted as ‘Sum’). We note again that this rule does not hold in general, and therefore, we proposed a power method to better approximate the sum-of-the-Kronecker products as a Kronecker product. We denote our approach as the ‘power method’ here.

First, we evaluate by generating positive definite matrices  $\mathbf{A} \in \mathbb{R}^{M \times M}$ ,  $\mathbf{G} \in \mathbb{R}^{N \times N}$  with  $M = N$ . One variation can be the size of the matrices from two to 20. We obtain them by sampling the elements from a standard normal distribution, then multiplying its transpose, and finally adding  $10^{-8} \cdot \mathbf{I}$ . We also scale the resulting matrix by a scale from  $10^{-5}$  to  $10^4$  and also use different weight decays  $10^{-10}$  to one within  $(\mathbf{A} \otimes \mathbf{G} + \gamma \mathbf{I})$ . For each size, scale, and weight decay, we use further 100 different random seeds. The results are shown in figure 3.3. In the plots, we show the mean and the 95% confidence interval of the relative Frobenius error for the two different approximations. Within these settings, we find that the power method has a relative Frobenius error of  $0.01020 \pm 0.03129$ . On the other hand, computing the sum of each factor as in (Ritter et al., 2018b) by  $(\mathbf{A} + \sqrt{\gamma} \mathbf{I}) \otimes (\mathbf{G} + \sqrt{\gamma} \mathbf{I})$ , has a relative Frobenius error of  $0.10167 \pm 0.22935$ . So, our experiments show that the power method yields at least one magnitude smaller in terms of the Frobenius error.

Furthermore, we also evaluate the sums-of-Kronecker-products for more than two matrices. Such evaluation is important for settings such as BPNN, where such computations are needed for each task. To do so, we compute the relative Frobenius error for  $\sum_{k=1}^K \mathbf{A}^k \otimes \mathbf{G}^k - \mathbf{A} \otimes \mathbf{G}$  with  $K$  ranging from two to nine. We use  $M = N = 5$  and sample the matrices similar to the aforementioned way:  $\mathbf{A}$  and  $\mathbf{G}$  (see figure 3.3). The baseline is again the ‘sum’. As in (Ritter et al., 2018a), we sum over more than two Kronecker-factored matrices, where the first matrix is an identity matrix scaled by the weight decay  $\gamma$ . The results are depicted in figure 3.3. We observed that the relative error is not only low with the power iteration method, but also the relative error of the baseline “sum” significantly increases. These results motivate the proposed method to compute the sums-of-Kronecker-products as two Kronecker factors.

### A.5.2. Implementation Details: Section 3.4.2

The implementation details for section 3.4.2 are as follows. Following Grosse & Martens (2016), we do not set a prior on the parameters of batch normalization. All images are normalized by the mean and standard deviation from the ImageNet data-set and resized to  $224 \times 224$ . We further use the Adam optimizer with batch size 4 and learning rate  $10^{-4}$ . We compare our method to PNNs with weight decays  $10^{-1}, 10^{-2}, \dots, 10^{-10}$ , PNNs using Monte Carlo dropout (Gal & Ghahramani, 2015) without weight decay with dropout probability 0.01. Moreover, the learned prior is compared to an isotropic Gaussian prior around zero and around the pre-trained weights on the ImageNet-1K data-set with weight decay  $10^{-8}$ . The temperature scaling is inferred after the training of each task with a coarse grid search.

### A.5.3. Implementation Details: Section 3.4.3

The implementation details for section 3.4.3 are as follows. Similar to the continual learning experiments (section 3.4.2), we do not set a prior for batch normalization. All images are resized to  $224 \times 224$  after normalization. For comparing isotropic priors and learned priors, we use the Adam optimizer with a batch size of 4 and a learning rate of  $10^{-4}$ . On the other hand, for other baselines, we tried four more learning rates:  $10^{-2}$ ,  $10^{-3}$ ,  $10^{-4}$  and  $10^{-5}$  in order to increase the few-shot learning performance. We did not use one-shot learning due to the validity of LA, but rather increased the number of evaluation points. While we used the software of [Tran et al. \(2022\)](#) for creating the few-shot learning dataset pairs. On top of that, we created separate validation sets from the remaining pool of data with a 1:1 ratio to the few-shot training set. For the baselines, we use a deep ensemble with five members. [Lakshminarayanan et al. \(2017\)](#) showed that for the therein presented classification experiments, the five members were able to closely match the performance of more members. The dropout rates in Monte Carlo dropout are the same as what is available. In all the experiments, we used 100 samples from the posterior for computing the predictive distribution.

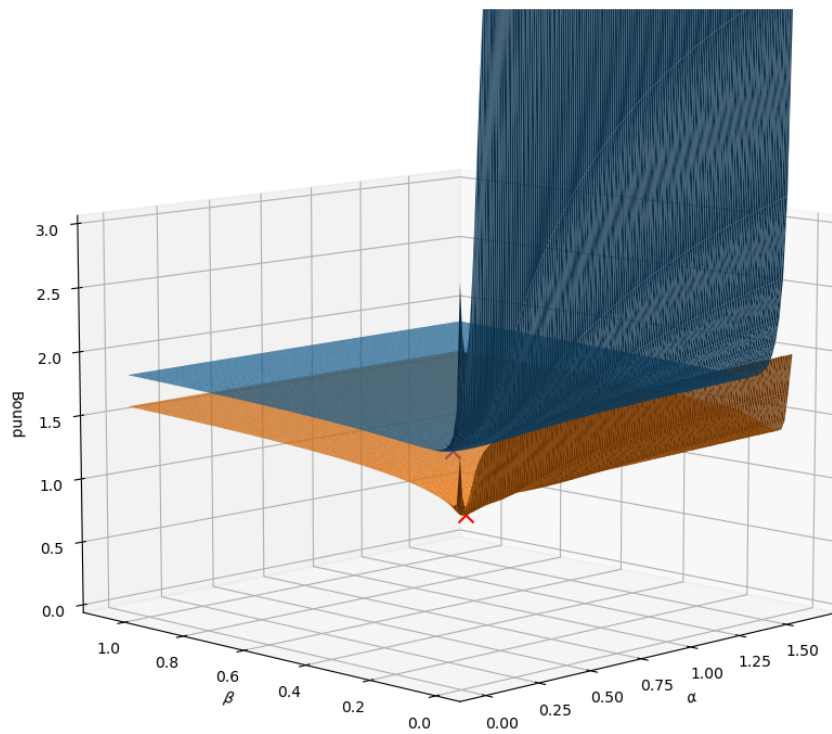
## A.6. Additional Experiments and Results

### A.6.1. Qualitative Analysis of the Approximate Upper Bound

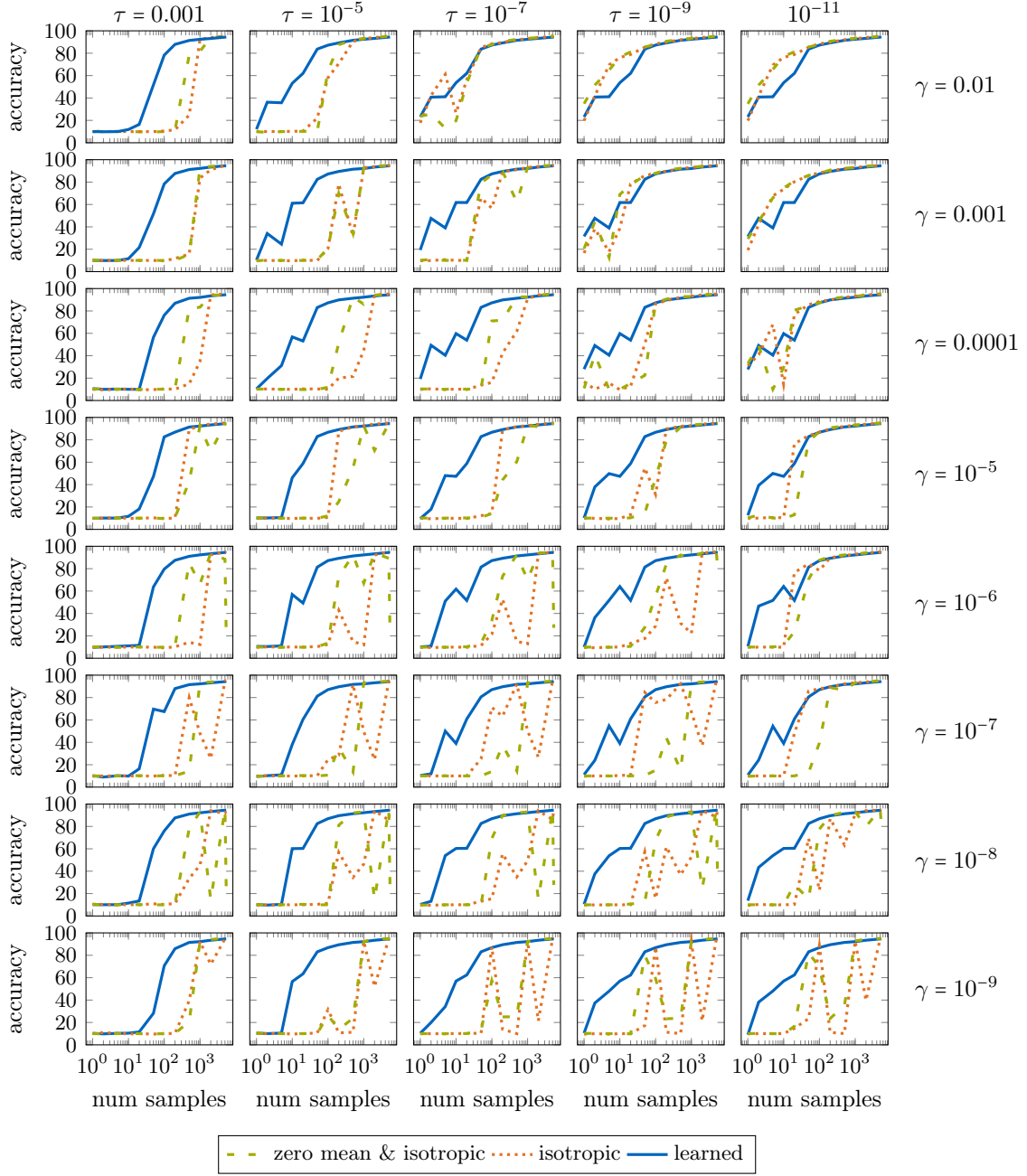
Here, we want to qualitatively analyze the quality of the upper bound and obtain further insights about their validity. To do so, we used the same setting as our ablation studies presented in section 3.4. One advantage is that the scale is small enough that we can directly compare the true PAC-Bayes bound and the proposed approximation to that bound. However, unlike other experiments, we share  $\alpha$  and  $\beta$  between layers. This was to meaningfully see the shape of the bound in 3D plots. The results are depicted in Figure A.1. Within this experiment, we observe that the true bound (orange) is tighter but follows qualitatively similar behavior as the proposed approximation (blue). In particular, we find that the optima (orange cross for a true bound and blue cross for approximation) are close to each other. Therefore, we believe that this data shows the quality of the proposed approximation at a small scale.

### A.6.2. Ablations in the Small-Data Regime

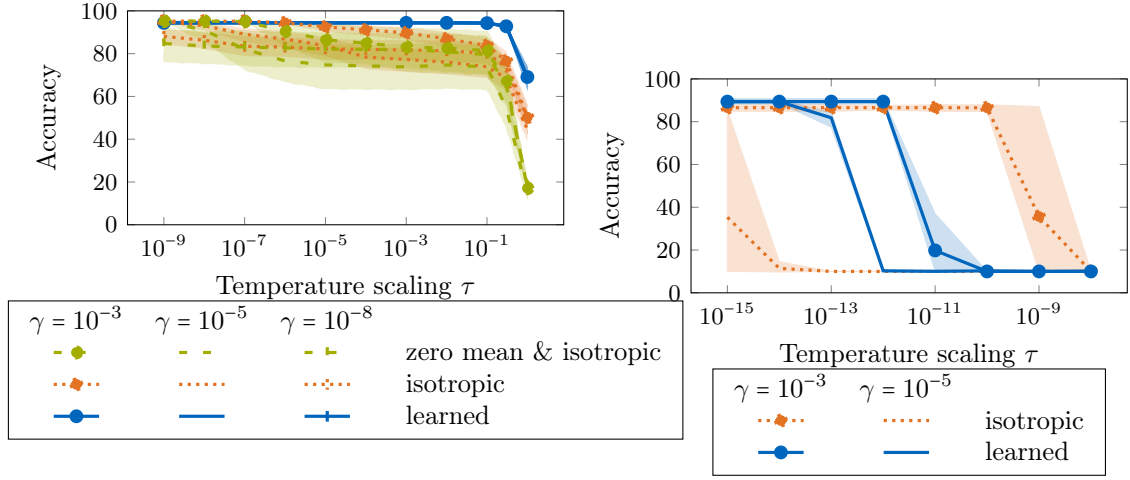
Next, we extend the ablation studies from section 3.4 in order to provide the influence of hyperparameters, namely temperature scaling and weight decays. To achieve this, we keep the same setting as the ablations again and plot for a specified range. Similarly again, as often done in the PAC-Bayes literature, we vary the availability of the data and compute the accuracy as a metric. The results can be found in Figure A.2. Importantly, this figure shows that the learned prior is in general more data-efficient. However, the results also show the conditions under which our method is effective. To explain, we observe that for small temperature scaling and large weight decays (top right), the performance of the learned prior deteriorates. The results are expected because, in the case of large weight decay, the isotropic prior becomes a more dominant term in the learned prior. Thus, the differences are reduced. On the other hand, for a small weight decay, BNNs become closer to deterministic. In that case, learning the prior only helps for small weight decay. Lastly,



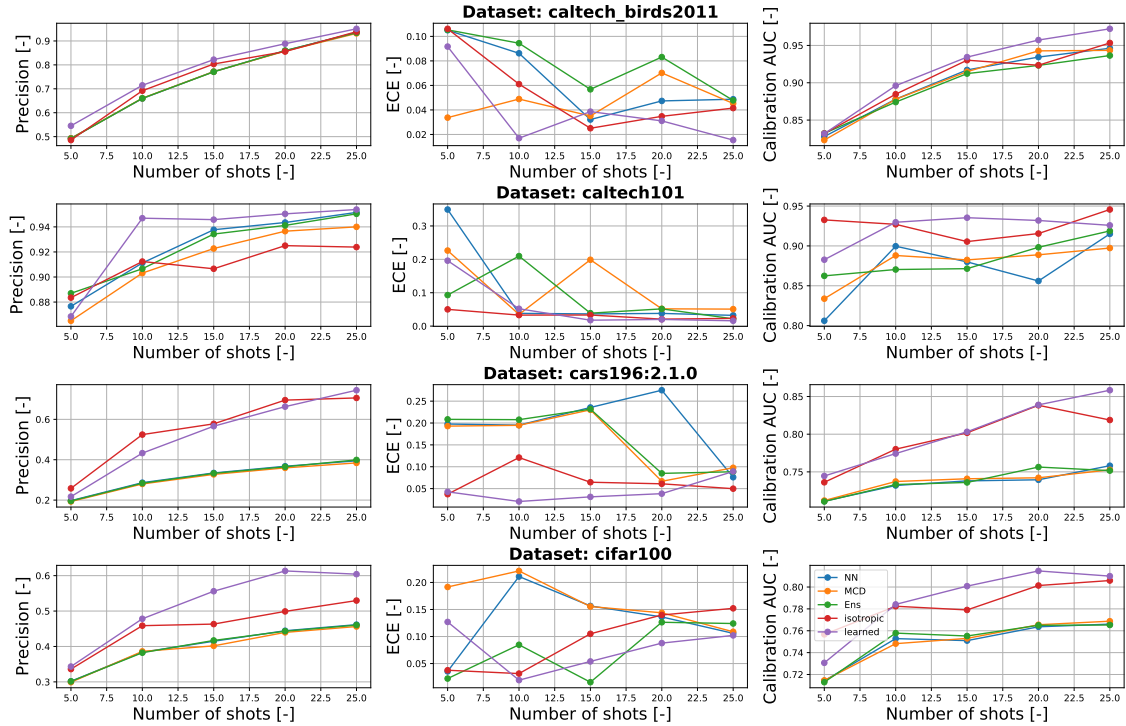
**Figure A.1.:** True bound (orange) Versus the proposed approximation (blue). The results show that the proposed approximation exhibits a similar shape to the true bound, providing qualitative insights about the proposed approximation. Smaller the bound, the better.



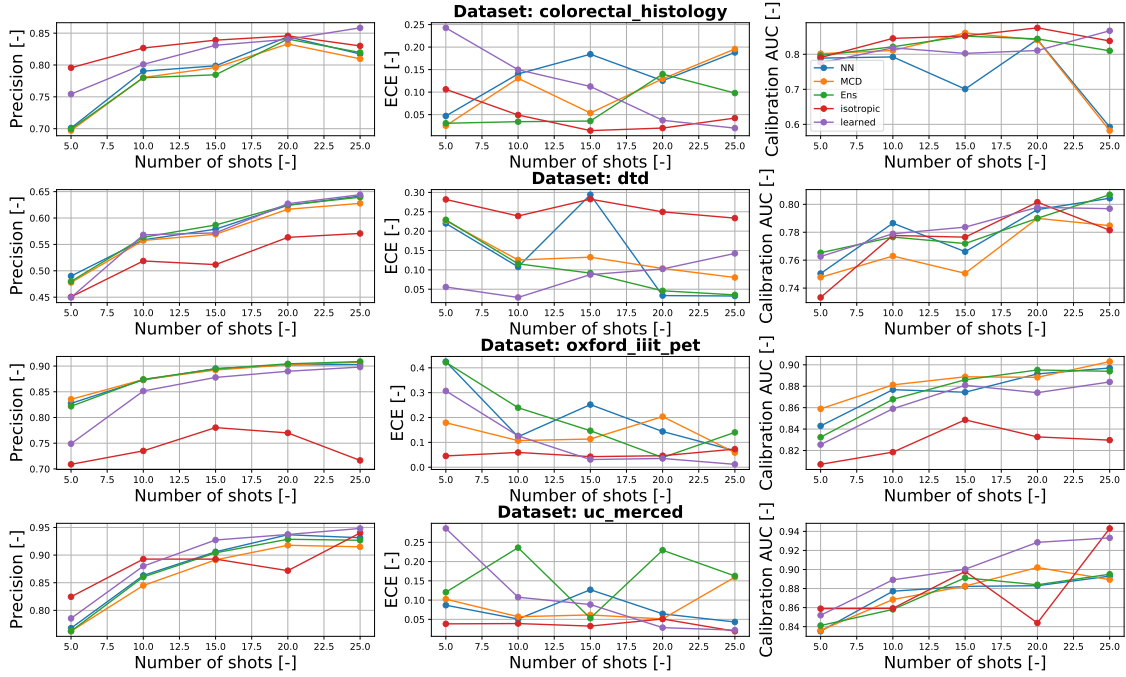
**Figure A.2.:** Extension of ablations in the small-data regime via varying two hyperparameters, namely temperature scaling and weight decays. The results empirically show conditions when the learned prior enables more data-efficient learning. Faster the increase in accuracy, the better.



**Figure A.3.:** Right: Extended cold posterior experiments with more weight decay variations. The learned prior produces more stable results with respect to variations in the weight decay. Left: Further validation with the prior learned from ImageNet. Learned prior from ImageNet and tested on CIFAR-10. The maximum reachable accuracy is higher with the learned prior. Learned prior also generates more stable results with respect to variations in weight decays.



**Figure A.4.:** Few-shot learning experiments. Results are presented for each dataset. The higher the better the accuracy and AUC measures, while the lower the better the ECE measures. The results show the high performance of our method in both uncertainty calibration and generalization.



**Figure A.5.:** Few-shot learning experiments. Results are presented for each dataset. The higher the better the accuracy and AUC measures, while the lower the better the ECE measures. The results show the high performance of our method in both uncertainty calibration and generalization.

we find that in a wide range of hyperparameter settings, the learned prior helps to learn more data efficiently when compared to isotropic priors.

### A.6.3. Further Cold Posterior Experiments

**Small-scale.** In section 3.4, we presented an empirical result on cold posterior effects. There, we showed that the cold posterior effect is not completely vanished, but is largely mitigated with the proposed learned prior. In this section, we aim to examine the stability of these results with respect to different weight decay settings. To this end, within the setting of the ablations, we examine variations of three weight decays. The results are shown in Figure A.3. In the experiments, we observe similar qualitative behavior, i.e., our learning-based prior largely mitigates the cold posterior effects when compared to the isotropic prior. We also ablate the prior with learned mean, which helps over purely zero mean isotropic Gaussian. However, incorporating the curvature, as in our method, further improves the results. Most importantly, our learned prior generates the least deviations as a result of the changes in the weight decay. Thus, we interpret the results that the learned priors are more stable with respect to changing the weight decay, while better mitigating the cold posterior effects.

**Large-scale.** Next, we further examine the cold posteriors within a large-scale experiment. To do so, we obtain the posteriors from ImageNet using the ResNet-50 architecture and use it as a priori in CIFAR-10. Using CIFAR-10, we obtain the posterior. We used a learning rate of  $1e-3$  and a batch size of eight. All other settings are kept the same as in the ablations of section 3.4. We do not evaluate the zero mean & isotropic prior but only the isotropic prior with the learned mean. This enables direct comparison of the methods, as we only vary the prior. The results are presented in Figure A.3. We observe the following.

First, we find that the cold posterior effect is strong for weight decay  $1e-3$ , whereas with the weight decay  $1e-5$ , the learned prior helps. The weight decay of  $1e-8$  did not produce reasonable results for all evaluated temperature scales (up to  $1e-15$ ). These results have been omitted from the plot for clarity of presentation. Finally, the reachable accuracy is higher for the learning-based priors than for the isotropic priors with the learned mean, while the influence of random seeds on deviations seems smaller.

#### A.6.4. Few-Shot Learning: Results per Dataset

In section 3.4, we have presented few-shot learning results, which were averaged in eight datasets. Now, we present the results per individual dataset. Hence, the results herein are an elaboration of section 3.4. Figures A.4 and A.5 represent the results, four different datasets each. We observe that in most datasets, the learned prior outperforms the isotropic priors as well as the presented baselines, in terms of precision and calibration measures. On the other hand, we also observe that the learned prior cannot uniformly outperform all other baselines across different datasets and metrics. For example, the colorectal histology dataset is one where the isotropic prior outperforms the learned prior, while in the oxford pet dataset, the standard BNNs and a deterministic neural network show stronger results in terms of precision and calibration AUC. We think that this is expected, as the prior learning methods assume the existence of relevant data and tasks from which to learn the prior from. Despite this limitation, since we find that the learned prior more often outperforms methods based on isotropic priors, our work shows the relevance of the prior learning methods, i.e., when there exist relevant data and tasks to learn from, there are potential performance advantages over relying on the isotropic priors.





## B.1. Derivations and Proofs

### B.1.1. Derivations of Low-Rank Sampling Computations

**Problem statement.** Let us assume that DNNs' parameter posterior is estimated with MND layer-wise. Without diagonal approximation or Kronecker factorization of the covariance matrix (or similarly information matrix or IM), the computational complexity of several operations, namely storage, inversion, and Cholesky decomposition, becomes intractable. For example, if there exists a layer with one million parameters, the storage of covariance alone scales quadratically. If we use the simple formula for back-of-the-envelope computations: total RAM for a  $N \times N$  double precision matrix requires  $N^2 * \frac{8}{10^9}$  gigabytes, storing a covariance matrix for  $N = 1000000$  requires 8000 gigabytes, which is computationally intractable for most modern computers. Consequently, cubic in cost operations, such as inversion or Cholesky decomposition, are not feasible in a naive strategy. This is a reason why the current approaches use diagonal approximation or Kronecker factorization of the covariance matrix if MND is the chosen posterior family. Following this statement, we list the memory-wise infeasible operations in detail: (a) naively extracting diagonal elements of  $(\mathbf{U}_A \otimes \mathbf{U}_G) \mathbf{\Lambda} (\mathbf{U}_A \otimes \mathbf{U}_G)^T$  and  $(\mathbf{U}_A \otimes \mathbf{U}_G) \mathbf{\Lambda}_{1:L} (\mathbf{U}_A \otimes \mathbf{U}_G)^T$ , naively storing, inverting, and Cholesky decomposing  $(\mathbf{U}_A \otimes \mathbf{U}_G) \mathbf{\Lambda}_{1:L} (\mathbf{U}_A \otimes \mathbf{U}_G)^T + \mathbf{D}$  (b) naively storing  $(\mathbf{U}_A \otimes \mathbf{U}_G)$  and performing LRA on  $(\mathbf{U}_A \otimes \mathbf{U}_G) \mathbf{\Lambda} (\mathbf{U}_A \otimes \mathbf{U}_G)^T$ . Our solution to the first two points is derived in this section while we have proposed an algorithm in the main manuscript to tackle the challenges of the last point. Table B.1 shows the main result on space complexity, where the proposed sparse information form of DNNs' posterior is also analyzed. Without resorting to mean-field approximations or matrix normal distribution, we show an alternative form of MND is also possible. Mathematical derivations we present below make this possible. Note that the inversion scheme considered is Gauss-Jordan elimination.

#### Diagonal Correction without Full Evaluation

Directly evaluating  $\mathbf{U}_A \otimes \mathbf{U}_G$  may not be computationally feasible for modern DNNs. Therefore, we derive the analytical form of the diagonal elements for  $(\mathbf{U}_A \otimes \mathbf{U}_G) \mathbf{\Lambda} (\mathbf{U}_A \otimes$

Operations	Space Complexity	
	Naive Strategy	Ours
Storage	$O(N^2)$	$O(L + na + mg)$
Inversion	$O(N^3)$	$O(L^3)$
Cholesky Decomposition	$O(N^3)$	$O(L^3)$

**Table B.1.:** Bounds on space complexity when compared to a naive strategy. We compare the several operations with a naive strategy to our presented derivations, which is the main results of our work. Here,  $L \ll N$  where  $L$  can be chosen with fidelity vs cost trade-off. This shows the sparse information form as a scalable Gaussian posterior family, providing alternatives to the diagonal covariance assumption or matrix normal distribution.

$U_G)^T$  without having to fully evaluate the Kronecker product. Let  $U_A \in \mathbb{R}^{n \times n}$  and  $U_G \in \mathbb{R}^{m \times m}$  be the square matrices.  $\Lambda \in \mathbb{R}^{mn \times mn}$  is a diagonal matrix by construction.  $V = U_A \otimes U_G \in \mathbb{R}^{mn \times mn}$  is a Kronecker product with elements  $v_{i,j}$  with  $i = m(\alpha - 1) + \gamma$  and  $j = m(\beta - 1) + \zeta$  (from the definition of the Kronecker product). Then, the diagonal entries of  $(U_A \otimes U_G)\Lambda(U_A \otimes U_G)^T$  can be computed as follows:

$$[(U_A \otimes U_G)\Lambda(U_A \otimes U_G)^T]_{ii} = \sum_{j=1}^{nm} (\mathbf{v}_{i,j} \sqrt{\Lambda_j})^2 \quad (\text{B.1})$$

**Derivation.** As a first step of the derivation, we express  $(U_A \otimes U_G)\Lambda(U_A \otimes U_G)^T$  in the following form:

$$\begin{aligned} (U_A \otimes U_G)\Lambda(U_A \otimes U_G)^T &= (U_A \otimes U_G)\Lambda^{\frac{1}{2}}\Lambda^{\frac{1}{2}}(U_A \otimes U_G)^T \\ &= [(U_A \otimes U_G)\Lambda^{\frac{1}{2}}] [(U_A \otimes U_G)\Lambda^{\frac{1}{2}}]^T \\ &= UU^T \end{aligned}$$

Then,  $\text{diag}(UU^T)_i = [UU^T]_{ii} = \sum_{j=1}^{nm} \mathbf{u}_{ij}^2$  by definition. Now, we let  $(U_A \otimes U_G)\Lambda^{\frac{1}{2}} = V\Lambda^{\frac{1}{2}}$  with  $\Lambda^{\frac{1}{2}}$  being again a diagonal matrix. Therefore,  $\mathbf{u}_{ij} = \mathbf{v}_{i,j} \sqrt{\Lambda_j}$  due to the multiplication with a diagonal matrix from a right-hand side. Substituting back these results in  $[(U_A \otimes U_G)\Lambda(U_A \otimes U_G)^T]_{ii} = \sum_{j=1}^{nm} (\mathbf{v}_{i,j} \sqrt{\Lambda_j})^2$  which completes the derivation. Formulating equation B.1 for the non-square matrices (which result after a low-rank approximation) such as  $U_A \in \mathbb{R}^{n \times a}$  and  $U_G \in \mathbb{R}^{m \times g}$  and paralleling this operation is rather trivial and hence, we omit this part of the derivation.

### Low Rank Sampler - Analytical Form

For a full Bayesian analysis, which is approximated by a Monte Carlo integration, sampling is a crucial operation for predicting uncertainty. We start by stating the problem.

**Problem statement.** Consider drawing samples  $\theta^{(t)} \in \mathbb{R}^{nm}$  from the proposed sparse information form:

$$\theta^{(t)} \sim \mathcal{N}^{-1}(\hat{\theta}_\eta, (U_A \otimes U_G)\Lambda_{1:L}(U_A \otimes U_G)^T + D) \quad (\text{B.2})$$

Drawing such samples from a covariance form of MND requires finding a symmetrical factor of the covariance matrix (e.g. Cholesky decomposition) which is cubic in cost  $O(N^3)$ . Even worse, when represented in an information form as in equation B.2, it requires first an inversion of the information matrix and then the computation of a symmetrical factor which

overall constitutes two operations of cost  $O(N^3)$ . Clearly, if  $N$  lies in a high dimension such as 1 million, even storing is obviously not feasible, let alone the sampling computations. Therefore, we need a sampling computation that (a) keeps the Kronecker structure while sampling so that first, the storage is memory-wise feasible, and then (b) the operations that require cubic cost such as inversion must be performed in the dimensions of low rank  $L$  instead of full parameter dimensions  $N$ . We provide the solution below.

**Analytical solution.** Let us define  $\mathbf{x}_0 \in \mathbb{R}^{mn}$  and  $\mathbf{X}_0 \in \mathbb{R}^{m \times n}$  as the samples from a standard Multivariate Normal Distribution in equation B.3 where we denote the following:  $\mathbf{0}_{nm} \in \mathbb{R}^{nm}$ ,  $\mathbf{I}_{mn} \in \mathbb{R}^{mn \times mn}$ ,  $\mathbf{0}_{n \times m} \in \mathbb{R}^{n \times m}$ ,  $\mathbf{I}_n \in \mathbb{R}^{n \times n}$ , and  $\mathbf{I}_m \in \mathbb{R}^{m \times m}$ . Note that these sampling operations are inexpensive.

$$\mathbf{x}_0 \sim N(\mathbf{0}_{nm}, \mathbf{I}_{nm}) \quad \text{or} \quad \mathbf{X}_0 \sim \mathcal{MN}(\mathbf{0}_{n \times m}, \mathbf{I}_n, \mathbf{I}_m). \quad (\text{B.3})$$

Furthermore, we denote  $\boldsymbol{\theta}^{(t)} \in \mathbb{R}^{mn}$ ,  $\hat{\boldsymbol{\theta}} \in \mathbb{R}^{mn}$  as a sample from equation B.2 and its mean as a vector, respectively. We also note that  $\boldsymbol{\Lambda}_{1:L} \in \mathbb{R}^{L \times L}$  and  $\mathbf{D} \in \mathbb{R}^{mn \times mn}$  are the low-ranked forms of the re-scaled eigenvalues and the diagonal correction term, as previously defined.  $\mathbf{U}_A \in \mathbb{R}^{n \times a}$  and  $\mathbf{U}_G \in \mathbb{R}^{m \times g}$  are the eigenvectors of the low-ranked eigen-basis, so that  $n \geq a$ ,  $m \geq g$ , and  $L = ag$ . Then, the samples of B.2 can be computed analytically as <sup>1</sup>:

$$\boldsymbol{\theta}^{(t)} = \hat{\boldsymbol{\theta}} + \mathbf{L}^\dagger \mathbf{x}_0 \quad \text{where,} \quad \mathbf{L}^\dagger = \mathbf{D}^{-\frac{1}{2}} \left( \mathbf{I}_{nm} - \mathbf{D}^{-\frac{1}{2}} (\mathbf{U}_A \otimes \mathbf{U}_G) \boldsymbol{\Lambda}_{1:L}^{\frac{1}{2}} (\mathbf{C}^{-1} + \mathbf{V}_s^T \mathbf{V}_s)^{-1} \boldsymbol{\Lambda}_{1:L}^{\frac{1}{2}} (\mathbf{U}_A \otimes \mathbf{U}_G)^T \mathbf{D}^{-\frac{1}{2}} \right). \quad (\text{B.4})$$

Firstly, the symmetrical factor  $\mathbf{L}^\dagger \in \mathbb{R}^{mn \times mn}$  in equation B.4 is a function of matrices that are feasible to store as they involve diagonal matrices or small matrices in a Kronecker structure. Furthermore,

$$\begin{aligned} \mathbf{V}_s &= \mathbf{D}^{-\frac{1}{2}} (\mathbf{U}_A \otimes \mathbf{U}_G) \boldsymbol{\Lambda}_{1:L}^{\frac{1}{2}} \\ \mathbf{C} &= \mathbf{A}_{\text{cholesky}}^{-T} (\mathbf{B}_{\text{cholesky}} - \mathbf{I}_L) \mathbf{A}_{\text{cholesky}}^{-1} \quad \text{with } \mathbf{A}_{\text{cholesky}} \text{ and } \mathbf{B}_{\text{cholesky}} \end{aligned} \quad (\text{B.5})$$

being the Cholesky decomposed matrices of  $\mathbf{V}_s^T \mathbf{V}_s \in \mathbb{R}^{L \times L}$  and  $\mathbf{V}_s^T \mathbf{V}_s + \mathbf{I}_L \in \mathbb{R}^{L \times L}$  such that:

$$\mathbf{A}_{\text{cholesky}} \mathbf{A}_{\text{cholesky}}^T = \mathbf{V}_s^T \mathbf{V}_s \quad \text{and} \quad \mathbf{B}_{\text{cholesky}} \mathbf{B}_{\text{cholesky}}^T = \mathbf{V}_s^T \mathbf{V}_s + \mathbf{I}_L. \quad (\text{B.6})$$

Consequently, the matrices in equation B.4 are defined as  $\mathbf{C} \in \mathbb{R}^{L \times L}$ ,  $(\mathbf{C}^{-1} + \mathbf{V}_s^T \mathbf{V}_s) \in \mathbb{R}^{L \times L}$ , and  $\mathbf{I}_L \in \mathbb{R}^{L \times L}$ . In this way, the two operations, namely Cholesky decomposition and inversion, that are cubic in cost  $O(N^3)$  are reduced to the low rank dimension  $L$  with complexity  $O(L^3)$ .

**Derivation.** Firstly, note that sampling from a standard multivariate Gaussian for  $\mathbf{x}_0$  or  $\mathbf{X}_0$  is computationally cheap (see equation B.3). Given a symmetrical factor for the covariance  $\boldsymbol{\Sigma} = \mathbf{L}^\dagger \mathbf{L}^{\dagger T}$  (e.g. by Cholesky decomposition), samples can be drawn via  $\hat{\boldsymbol{\theta}} + \mathbf{L}^\dagger \mathbf{x}_0$  as depicted in equation B.4. Our derivation involves finding such a symmetrical factor for the given form of the covariance matrix while exploring the Kronecker structure for the sampling computations so that the space complexity is bounded to  $O(L^3)$  instead of  $O(N^3)$ .

Let us first reformulate the covariance (inverse of the information matrix) as follows.

<sup>1</sup>We show how the Kronecker structure of  $\mathbf{L}^\dagger$  can be exploited to compute  $\mathbf{L}^\dagger \mathbf{x}_0$  in the derivation only.

$$\begin{aligned}
 \Sigma &= \left( (U_A \otimes U_G) \Lambda_{1:L} (U_A \otimes U_G)^T + D \right)^{-1} \\
 &= \left[ D^{\frac{1}{2}} \left( D^{-\frac{1}{2}} (U_A \otimes U_G) \Lambda_{1:L}^{\frac{1}{2}} \Lambda_{1:L}^{\frac{1}{2}} (U_A \otimes U_G)^T D^{-\frac{1}{2}} + I_{nm} \right) D^{\frac{1}{2}} \right]^{-1} \\
 &= D^{-\frac{1}{2}} \left[ \left( D^{-\frac{1}{2}} (U_A \otimes U_G) \Lambda_{1:L}^{\frac{1}{2}} \right) \left( D^{-\frac{1}{2}} (U_A \otimes U_G) \Lambda_{1:L}^{\frac{1}{2}} \right)^T + I_{nm} \right]^{-1} D^{-\frac{1}{2}} \\
 &= D^{-\frac{1}{2}} \left[ V_s V_s^T + I_{nm} \right]^{-1} D^{-\frac{1}{2}}.
 \end{aligned} \tag{B.7}$$

Here, we define:  $V_s = D^{-\frac{1}{2}} (U_A \otimes U_G) \Lambda_{1:L}^{\frac{1}{2}}$ . Now, a symmetrical factor for  $\Sigma = L^\dagger L^{\dagger T}$  can be found by exploiting the above structure. We let  $U_s$  be a symmetrical factor for  $V_s V_s^T + I_{nm}$  so that  $L^\dagger = D^{-\frac{1}{2}} W^{c-1}$  is the symmetrical factor of  $\Sigma$ . Following the work of [Ambikasaran & O'Neil \(2014\)](#) the symmetrical factor  $U_s$  can be found using equations:

$$\begin{aligned}
 U_s &= I_{nm} + V_s C V_s^T \\
 C &= A_{\text{cholesky}}^{-T} (B_{\text{cholesky}} - I_L) A_{\text{cholesky}}^{-1}.
 \end{aligned} \tag{B.8}$$

Note that A and B are Cholesky decomposed matrices of  $V_s^T V_s \in \mathbb{R}^{L \times L}$  and  $V_s^T V_s + I_L \in \mathbb{R}^{L \times L}$  respectively. As a first result, this operation is bounded by complexity  $O(L^3)$  instead of the full parameter dimension  $N$ . Calculations of  $V_s^T V_s$  can also be performed in iterations. Now the symmetrical factor for  $\Sigma$  can be expressed as follows:

$$\begin{aligned}
 L^\dagger &= D^{-\frac{1}{2}} U_s^{-1} = D^{-\frac{1}{2}} (I_{nm} + V_s C V_s^T)^{-1} \\
 &= D^{-\frac{1}{2}} \left( I_{nm} - V_s (C^{-1} + V_s^T V_s)^{-1} V_s^T \right).
 \end{aligned} \tag{B.9}$$

Woodbury's Identity is used here. Now, by substitution:

$$\begin{aligned}
 \theta^{(t)} &= \hat{\theta} + L^\dagger x_0 \quad \text{where,} \\
 L^\dagger &= D^{-\frac{1}{2}} \left( I_{nm} - V_s (C^{-1} + V_s^T V_s)^{-1} V_s^T \right) \\
 &= D^{-\frac{1}{2}} \left( I_{nm} - D^{-\frac{1}{2}} (U_A \otimes U_G) \Lambda_{1:L}^{\frac{1}{2}} (C^{-1} + V_s^T V_s)^{-1} \Lambda_{1:L}^{\frac{1}{2}} (U_A \otimes U_G)^T D^{-\frac{1}{2}} \right).
 \end{aligned} \tag{B.10}$$

This completes the derivation of equation B.4. As a result, the inversion operation is bounded by complexity  $O(L^3)$ . Furthermore, the derivation constitutes smaller matrices  $U_A$  and  $U_G$  or diagonal matrices  $D$  and  $I_{mn}$ , which can be stored as vectors. In short, the complexity has significantly decreased.

Now we further derive computations that exploit rules of Kronecker products. Consider:

$$L^\dagger x_0 = D^{-\frac{1}{2}} \left( I_{nm} - D^{-\frac{1}{2}} (U_A \otimes U_G) \Lambda_{1:L}^{\frac{1}{2}} (C^{-1} + V_s^T V_s)^{-1} \Lambda_{1:L}^{\frac{1}{2}} (U_A \otimes U_G)^T D^{-\frac{1}{2}} \right) x_0. \tag{B.11}$$

Then, it follows by defining the inverted matrix  $L_c = (C^{-1} + V_s^T V_s)^{-1} \in \mathbb{R}^{L \times L}$  with a cost  $O(L^3)$ :

$$\mathbf{L}^\dagger \mathbf{x}_0 = \mathbf{D}^{-\frac{1}{2}} \left( \mathbf{I}_{nm} - \mathbf{D}^{-\frac{1}{2}} (\mathbf{U}_A \otimes \mathbf{U}_G) \mathbf{\Lambda}_{1:L}^{\frac{1}{2}} \mathbf{L}_c \mathbf{\Lambda}_{1:L}^{\frac{1}{2}} (\mathbf{U}_A \otimes \mathbf{U}_G)^T \mathbf{D}^{-\frac{1}{2}} \right) \mathbf{x}_0. \quad (\text{B.12})$$

We further reduce this by evaluating  $\mathbf{D}^{-\frac{1}{2}}$  and defining  $\mathbf{X}_D^l = \mathbf{D}^{-\frac{1}{2}} \mathbf{x}_0 \in \mathbb{R}^{mn}$  and  $\mathbf{P}_c = \mathbf{\Lambda}_{1:L}^{\frac{1}{2}} \mathbf{L}_c \mathbf{\Lambda}_{1:L}^{\frac{1}{2}} \in \mathbb{R}^{L \times L}$ . Note that this multiplication operation is memory-wise feasible.

$$\mathbf{L}^\dagger \mathbf{x}_0 = \mathbf{X}_D^l - \left( \mathbf{D}^{-1} (\mathbf{U}_A \otimes \mathbf{U}_G) \mathbf{P}_c (\mathbf{U}_A \otimes \mathbf{U}_G)^T \mathbf{X}_D^l \right). \quad (\text{B.13})$$

Now, we map  $\mathbf{X}_D^l$  to matrix normal distribution by an  $\text{unvec}(\cdot)$  operation so that  $\mathbf{X}_D^s = \text{unvec}(\mathbf{X}_D^l) \in \mathbb{R}^{m \times n}$  or equivalently  $\mathbf{X}_D^l = \text{vec}(\mathbf{X}_D^s)$ . Using a widely known relation for Kronecker product that is -  $(\mathbf{U}_A \otimes \mathbf{U}_G)^T \text{vec}(\mathbf{X}_D^s) = \text{vec}(\mathbf{U}_G^T \mathbf{X}_D^s \mathbf{U}_A)$ , it follows:

$$\mathbf{L}^\dagger \mathbf{x}_0 = \mathbf{X}_D^l - \left( \mathbf{D}^{-1} (\mathbf{U}_A \otimes \mathbf{U}_G) \mathbf{P}_c \text{vec}(\mathbf{U}_G^T \mathbf{X}_D^s \mathbf{U}_A) \right). \quad (\text{B.14})$$

Note that matrix multiplication is performed with small matrices. Repeating a similar procedure as above, we obtain the equation below for  $\mathbf{X}_P^s = \mathbf{P}_c (\mathbf{U}_A \otimes \mathbf{U}_G)^T \mathbf{X}_D^l$ ,

$$\begin{aligned} \mathbf{L}^\dagger \mathbf{x}_0 &= \mathbf{X}_D^l - \left( \mathbf{D}^{-1} (\mathbf{U}_A \otimes \mathbf{U}_G) \mathbf{X}_P^s \right) \\ &= \mathbf{X}_D^l - \left( \mathbf{D}^{-1} \text{vec}(\mathbf{U}_G \mathbf{X}_P^s \mathbf{U}_A^T) \right). \end{aligned} \quad (\text{B.15})$$

This completes the derivation. In summary, we presented a new derivation to sample from equation B.2, a low-rank and information formulation of MND. This analytical solution ensures (a)  $O(N^3) \gg O(L^3)$  for Cholesky decomposition, (b)  $O(N^3) \gg O(L^3)$  for a matrix inversion, (c) storage of small matrices  $\mathbf{U}_G$ ,  $\mathbf{U}_A$ , a diagonal matrix  $\mathbf{D}$  and identity matrices, and finally (d) matrix multiplications that only involve these matrices. This is a direct benefit of our proposed LRA that preserves Kronecker structure in eigenvectors. Furthermore, this result shows the sparse information form as a new scalable Gaussian posterior family for approximate Bayesian inference.

### B.1.2. Proof of Lemma 4.3.1

**Proposition B.1.1.** *Let  $\mathbf{F} \in \mathbb{R}^{N \times N}$  be the real information matrix, and let  $\mathbf{F}_{\text{inf}} \in \mathbb{R}^{N \times N}$  and  $\hat{\mathbf{F}}_{\text{inf}} \in \mathbb{R}^{N \times N}$  be our estimates of it with rank  $d$  and  $k$  such that  $k < d$ . Their diagonal entries are equal; that is,  $\mathbf{F}_{ii} = \mathbf{F}_{\text{inf}ii} = \hat{\mathbf{F}}_{\text{inf}ii}$  for all  $i = 1, \dots, N$ .*

*Proof.* The proof trivially follows from the definitions of  $\mathbf{F} \in \mathbb{R}^{N \times N}$ ,  $\mathbf{F}_{\text{inf}} \in \mathbb{R}^{N \times N}$  and  $\hat{\mathbf{F}}_{\text{inf}} \in \mathbb{R}^{N \times N}$ . As the exact Fisher is an expectation on outer products of back-propagated gradients, its diagonal entries equal  $\mathbf{F}_{ii} = \mathbb{E}[\mathcal{D}\theta_i^2]$  for all  $i = 1, 2, \dots, N$ .

For a full ranked information matrix (IM)  $\mathbf{F}_{\text{inf}}$ , substituting  $\mathbf{D}_{ii} = \mathbb{E}[\mathcal{D}\theta_i^2] - \sum_{j=1}^{nm} (\mathbf{v}_{i,j} \sqrt{\mathbf{\Lambda}_j})^2$  with  $\sum_{j=1}^{nm} (\mathbf{v}_{i,j} \sqrt{\mathbf{\Lambda}_j})^2 = (\mathbf{U}_A \otimes \mathbf{U}_G) \mathbf{\Lambda} (\mathbf{U}_A \otimes \mathbf{U}_G)^T_{ii}$  results in equation B.16 for all  $i = 1, 2, \dots, N$ ,

$$\begin{aligned}
 \mathbf{F}_{\text{inf}ii} &= (\mathbf{U}_A \otimes \mathbf{U}_G) \mathbf{\Lambda} (\mathbf{U}_A \otimes \mathbf{U}_G)_{ii}^T + \mathbf{D}_{ii} \\
 &= (\mathbf{U}_A \otimes \mathbf{U}_G) \mathbf{\Lambda} (\mathbf{U}_A \otimes \mathbf{U}_G)_{ii}^T + \mathbb{E}[\mathcal{D}\theta_i^2] \\
 &\quad - (\mathbf{U}_A \otimes \mathbf{U}_G) \mathbf{\Lambda} (\mathbf{U}_A \otimes \mathbf{U}_G)_{ii}^T \\
 &= \mathbb{E}[\mathcal{D}\theta_i^2].
 \end{aligned} \tag{B.16}$$

Similarly, we substitute  $\hat{\mathbf{D}}_{ii} = \mathbb{E}[\mathcal{D}\theta_i^2] - \sum_{j=1}^L (\hat{\mathbf{v}}_{i,j} \sqrt{\mathbf{\Lambda}_{1:L}})^2$  with  $\sum_{j=1}^L (\hat{\mathbf{v}}_{i,j} \sqrt{\mathbf{\Lambda}_{1:L}})^2 = (\mathbf{U}_A \otimes \mathbf{U}_G) \mathbf{\Lambda}_{1:L} (\mathbf{U}_A \otimes \mathbf{U}_G)_{ii}^T$ , which results in equation B.17 for all  $i = 1, 2, \dots, N$ ,

$$\begin{aligned}
 \hat{\mathbf{F}}_{\text{inf}ii} &= (\mathbf{U}_A \otimes \mathbf{U}_G) \mathbf{\Lambda}_{1:L} (\mathbf{U}_A \otimes \mathbf{U}_G)_{ii}^T + \mathbf{D}_{ii} \\
 &= (\mathbf{U}_A \otimes \mathbf{U}_G) \mathbf{\Lambda}_{1:L} (\mathbf{U}_A \otimes \mathbf{U}_G)_{ii}^T + \mathbb{E}[\mathcal{D}\theta_i^2] \\
 &\quad - (\mathbf{U}_A \otimes \mathbf{U}_G) \mathbf{\Lambda}_{1:L} (\mathbf{U}_A \otimes \mathbf{U}_G)_{ii}^T \\
 &= \mathbb{E}[\mathcal{D}\theta_i^2].
 \end{aligned} \tag{B.17}$$

Therefore, we have  $\mathbf{F}_{ii} = \mathbf{F}_{\text{inf}ii} = \hat{\mathbf{F}}_{\text{inf}ii}$  for all  $i = 1, 2, \dots, N$ .  $\square$

**Lemma 4.3.1.** *Let  $\mathbf{F}$  be the real information matrix, and let  $\mathbf{F}_{\text{inf}}$  and  $\mathbf{F}_{\text{efb}}$  be the INF and EFB estimates of it, respectively. It is guaranteed to have  $\|\mathbf{F} - \mathbf{F}_{\text{efb}}\|_F \geq \|\mathbf{F} - \mathbf{F}_{\text{inf}}\|_F$ .*

*Proof.* Let  $e^2 = \|\mathbf{A} - \mathbf{B}\|_F^2$  define a squared Frobenius norm of error between the two matrices  $\mathbf{A} \in \mathbb{R}^{N \times N}$  and  $\mathbf{B} \in \mathbb{R}^{N \times N}$ . Now,  $e^2$  can be formulated as,

$$\begin{aligned}
 e_b^2 &= \|\mathbf{A} - \mathbf{B}\|_F^2 \\
 &= \sum_i (\mathbf{A} - \mathbf{B})_{ii}^2 + \sum_i \sum_{j \neq i} (\mathbf{A} - \mathbf{B})_{ij}^2
 \end{aligned} \tag{B.18}$$

The first term of equation B.18 belongs to errors of diagonal entries in B w.r.t A, whilst the second term is due to the off-diagonal entries.

Now, it follows that,

$$\begin{aligned}
 \|\mathbf{F} - \mathbf{F}_{\text{efb}}\|_F &\geq \|\mathbf{F} - \mathbf{F}_{\text{inf}}\|_F \\
 e_{\text{efb}}^2 &\geq e_{\text{inf}}^2 \\
 \sum_i (\mathbf{F} - \mathbf{F}_{\text{efb}})_{ii}^2 + \sum_i \sum_{j \neq i} (\mathbf{F} - \mathbf{F}_{\text{efb}})_{ij}^2 &\geq \sum_i (\mathbf{F} - \mathbf{F}_{\text{inf}})_{ii}^2 + \sum_i \sum_{j \neq i} (\mathbf{F} - \mathbf{F}_{\text{inf}})_{ij}^2 \\
 \sum_i (\mathbf{F} - \mathbf{F}_{\text{efb}})_{ii}^2 + \sum_i \sum_{j \neq i} (\mathbf{F} - \mathbf{F}_{\text{efb}})_{ij}^2 &\geq \sum_i \sum_{j \neq i} (\mathbf{F} - \mathbf{F}_{\text{inf}})_{ij}^2 \\
 \sum_i (\mathbf{F} - \mathbf{F}_{\text{efb}})_{ii}^2 + \sum_i \sum_{j \neq i} (\mathbf{F} - \mathbf{F}_{\text{efb}})_{ij}^2 &\geq \sum_i \sum_{j \neq i} (\mathbf{F} - \mathbf{F}_{\text{efb}})_{ij}^2
 \end{aligned}$$

Note that  $\sum_i (\mathbf{F} - \mathbf{F}_{\text{inf}})_{ii}^2 = 0$  using the first proposition. Furthermore,  $\sum_i \sum_{j \neq i} (\mathbf{F} - \mathbf{F}_{\text{inf}})_{ij}^2 = \sum_i \sum_{j \neq i} (\mathbf{F} - \mathbf{F}_{\text{efb}})_{ij}^2$  since by definition,  $\mathbf{F}_{\text{efb}}$  and  $\mathbf{F}_{\text{inf}}$  have the same off-diagonal terms.  $\square$

**Corollary 4.3.2.** *Let  $\mathbf{F}_{\text{kfac}}$  and  $\mathbf{F}_{\text{inf}}$  be KFAC and our estimates of the real information matrix  $\mathbf{F}$  respectively. Then, it is guaranteed to have  $\|\mathbf{F} - \mathbf{F}_{\text{kfac}}\|_F \geq \|\mathbf{F} - \mathbf{F}_{\text{inf}}\|_F$ .*

For interested readers, find the proof  $\|\mathbf{F} - \mathbf{F}_{\text{kfac}}\|_F \geq \|\mathbf{F} - \mathbf{F}_{\text{efb}}\|_F$  in [George et al. \(2018\)](#).



### B.1.3. Proof of Lemma 4.3.3

**Lemma 4.3.3.** *Let  $\mathbf{F}$  be the real Fisher information matrix, and let  $\hat{\mathbf{F}}_{\text{inf}}$ ,  $\mathbf{F}_{\text{efb}}$  and  $\mathbf{F}_{\text{kfac}}$  be the low rank INF, EFB, and KFAC estimates of it, respectively. Then, it is guaranteed to have  $\|\text{diag}(\mathbf{F}) - \text{diag}(\mathbf{F}_{\text{efb}})\|_F \geq \|\text{diag}(\mathbf{F}) - \text{diag}(\hat{\mathbf{F}}_{\text{inf}})\|_F = 0$  and  $\|\text{diag}(\mathbf{F}) - \text{diag}(\mathbf{F}_{\text{kfac}})\|_F \geq \|\text{diag}(\mathbf{F}) - \text{diag}(\hat{\mathbf{F}}_{\text{inf}})\|_F = 0$ . Furthermore, if the eigenvalues of  $\hat{\mathbf{F}}_{\text{inf}}$  contain all non-zero eigenvalues of  $\mathbf{F}_{\text{inf}}$ , it follows:  $\|\mathbf{F} - \mathbf{F}_{\text{efb}}\|_F \geq \|\mathbf{F} - \hat{\mathbf{F}}_{\text{inf}}\|_F$ .*

*Proof.* The first part follows from proposition B.1.1 which states that for all the elements  $i$ ,  $\mathbf{F}_{ii} = \hat{\mathbf{F}}_{\text{inf}}$ ,  $\|\text{diag}(\mathbf{F}) - \text{diag}(\mathbf{F}_{\text{efb}})\|_F \geq \|\text{diag}(\mathbf{F}) - \text{diag}(\hat{\mathbf{F}}_{\text{inf}})\|_F = 0$  and  $\|\text{diag}(\mathbf{F}) - \text{diag}(\mathbf{F}_{\text{kfac}})\|_F \geq \|\text{diag}(\mathbf{F}) - \text{diag}(\hat{\mathbf{F}}_{\text{inf}})\|_F = 0$ . This results from the design of the method, in which we correct the diagonals in parameter space after LRA.

For the second part of the proof, let's recap Wely's idea on eigenvalue perturbation, which states that removing zero eigenvalues does not affect the approximation error in terms of Frobenius norm. This then implies that off-diagonal elements of  $\hat{\mathbf{F}}_{\text{inf}}$  and  $\mathbf{F}_{\text{efb}}$  are equivalent. Then:

$$\begin{aligned} \|\mathbf{F} - \mathbf{F}_{\text{efb}}\|_F &\geq \|\mathbf{F} - \hat{\mathbf{F}}_{\text{inf}}\|_F \\ e_{\text{efb}}^2 &\geq e_{\text{inf}}^2 \\ \sum_i (\mathbf{F} - \mathbf{F}_{\text{efb}})_{ii}^2 + \sum_i \sum_{j \neq i} (\mathbf{F} - \mathbf{F}_{\text{efb}})_{ij}^2 &\geq \sum_i (\mathbf{F} - \hat{\mathbf{F}}_{\text{inf}})_{ii}^2 + \sum_i \sum_{j \neq i} (\mathbf{F} - \hat{\mathbf{F}}_{\text{inf}})_{ij}^2 \\ \sum_i (\mathbf{F} - \mathbf{F}_{\text{efb}})_{ii}^2 + \sum_i \sum_{j \neq i} (\mathbf{F} - \mathbf{F}_{\text{efb}})_{ij}^2 &\geq \sum_i \sum_{j \neq i} (\mathbf{F} - \hat{\mathbf{F}}_{\text{inf}})_{ij}^2 \\ \sum_i (\mathbf{F} - \mathbf{F}_{\text{efb}})_{ii}^2 + \sum_i \sum_{j \neq i} (\mathbf{F} - \mathbf{F}_{\text{efb}})_{ij}^2 &\geq \sum_i \sum_{j \neq i} (\mathbf{F} - \mathbf{F}_{\text{efb}})_{ij}^2 \end{aligned}$$

Again,  $\sum_i (\mathbf{F} - \hat{\mathbf{F}}_{\text{inf}})_{ii}^2 = 0$  according to proposition 1 for all the elements  $i$ , which completes the proof.  $\square$

### B.1.4. Proof of Lemma 4.3.4

**Lemma 4.3.4.** *The low rank matrix  $\hat{\Sigma} = ((\mathbf{U}_A \otimes \mathbf{U}_G) \Lambda_{1:L} (\mathbf{U}_A \otimes \mathbf{U}_G)^T + \mathbf{D})^{-1} \in \mathbb{R}^{N \times N}$  is a non-degenerate covariance matrix if the diagonal correction matrix  $\mathbf{D}$  and LRA  $(\mathbf{U}_A \otimes \mathbf{U}_G) \Lambda_{1:L} (\mathbf{U}_A \otimes \mathbf{U}_G)^T$  are both symmetric and positive definite. This condition is satisfied if  $(\mathbf{U}_A \otimes \mathbf{U}_G) \Lambda_{1:L} (\mathbf{U}_A \otimes \mathbf{U}_G)_{ii}^T < \mathbb{E}[\mathcal{D}\theta_i^2]$  for all  $i \in \{1, 2, \dots, N\}$  and with  $\Lambda_{1:L} \notin 0$ .*

*Proof.* Let us first rewrite  $\hat{\mathbf{F}}_{\text{inf}} = (\mathbf{U}_A \otimes \mathbf{U}_G) \Lambda_{1:L} (\mathbf{U}_A \otimes \mathbf{U}_G)^T + \mathbf{D}$  in the following form.

$$\begin{aligned} (\mathbf{U}_A \otimes \mathbf{U}_G) \Lambda_{1:L} (\mathbf{U}_A \otimes \mathbf{U}_G)^T + \mathbf{D} &= \\ (\mathbf{U}_A \otimes \mathbf{U}_G) \Lambda_{1:L}^{\frac{1}{2}} \Lambda_{1:L}^{\frac{1}{2}} (\mathbf{U}_A \otimes \mathbf{U}_G)^T + \mathbf{D} & \\ = \left[ (\mathbf{U}_A \otimes \mathbf{U}_G) \Lambda_{1:L}^{\frac{1}{2}} \right] \left[ (\mathbf{U}_A \otimes \mathbf{U}_G) \Lambda_{1:L}^{\frac{1}{2}} \right]^T + \mathbf{D} & \quad (\text{B.19}) \\ = \mathbf{U} \mathbf{U}^T + \mathbf{D} & \end{aligned}$$

Now, if  $\mathbf{D}$  and  $(\mathbf{U}_A \otimes \mathbf{U}_G) \Lambda_{1:L} (\mathbf{U}_A \otimes \mathbf{U}_G)^T$  are both symmetric and positive definite, it follows that for an arbitrary vector  $\mathbf{v} \in \mathbb{R}^d$ ,  $\mathbf{v}^T \mathbf{U} \mathbf{U}^T \mathbf{v} > 0$  as eigenvalues  $R_i > 0$  by construction. Furthermore,  $\mathbf{v}^T \mathbf{D} \mathbf{v} > 0$  also holds by the definition of positive definiteness. Therefore, we have  $\mathbf{v}^T (\mathbf{U} \mathbf{U}^T + \mathbf{D}) \mathbf{v} = \mathbf{v}^T \mathbf{U} \mathbf{U}^T \mathbf{v} + \mathbf{v}^T \mathbf{D} \mathbf{v} > 0$  which leads to the proof that  $\mathbf{F}_{\text{inf}}$  is positive definite if  $\mathbf{D}$  and  $(\mathbf{U}_A \otimes \mathbf{U}_G) \Lambda_{1:L} (\mathbf{U}_A \otimes \mathbf{U}_G)^T$  are both symmetric and positive definite. As this results in a non-degenerate IM, the covariance  $\Sigma$  is non-degenerate as well.

Trivially following the definition of  $\mathbf{D}_{ii} = \mathbb{E}[\mathcal{D}\theta_i^2] - (\mathbf{U}_A \otimes \mathbf{U}_G)\mathbf{\Lambda}(\mathbf{U}_A \otimes \mathbf{U}_G)^T$ ,  $\mathbf{D}_{ii} > 0$  for all  $i$  when  $(\mathbf{U}_A \otimes \mathbf{U}_G)\mathbf{\Lambda}_{1:L}(\mathbf{U}_A \otimes \mathbf{U}_G)^T < \mathbb{E}[\mathcal{D}\theta_i^2]$ . Again, by the definition of  $\mathbf{\Lambda}_{ii} = \mathbb{E}[(V^T \mathcal{D}\theta)_i^2] \geq 0$ ,  $\mathbf{\Lambda}_{1:L}$  containing no zero eigenvalues results in the positive definite matrix  $(\mathbf{U}_A \otimes \mathbf{U}_G)\mathbf{\Lambda}_{1:L}(\mathbf{U}_A \otimes \mathbf{U}_G)^T$ , which completes the proof.  $\square$

### B.1.5. Proof of Lemma 4.3.5

**Lemma 4.3.5.** *Let  $\mathbf{F} \in \mathbb{R}^{N \times N}$  be the real Fisher information matrix, and let  $\hat{\mathbf{F}}_{1:K}^{\text{top}} \in \mathbb{R}^{N \times N}$ ,  $\hat{\mathbf{F}}_{1:L}^{\text{top}} \in \mathbb{R}^{N \times N}$  and  $\hat{\mathbf{F}}_{1:L} \in \mathbb{R}^{N \times N}$  be the low-rank estimates of  $\mathbf{F}$  of EFB obtained by preserving top  $K$ ,  $L$  and top  $K$  plus additional  $J$ , resulting in  $L$  eigenvalues. Here, we define  $K < L$ . Then, the approximation error of  $\hat{\mathbf{F}}_{1:L}$  is as follows:  $\|\mathbf{F} - \hat{\mathbf{F}}_{1:L}^{\text{top}}\|_F \geq \|\mathbf{F} - \hat{\mathbf{F}}_{1:L}\|_F \geq \|\mathbf{F} - \hat{\mathbf{F}}_{1:K}^{\text{top}}\|_F$ .*

*Proof.* From the definition,  $(\mathbf{U}_A \otimes \mathbf{U}_G)\mathbf{\Lambda}(\mathbf{U}_A \otimes \mathbf{U}_G)^T = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^T$  is positive semi-definite as  $\mathbf{\Lambda}_{ii} = \mathbb{E}[(V^T \mathcal{D}\theta)_i^2] \geq 0$  for all elements  $i$  and  $\mathbf{V}\mathbf{V}^T = \mathbf{I}$  with  $\mathbf{I}$  as an identity matrix (orthogonality). Naturally, low rank approximations  $(\mathbf{U}_A \otimes \mathbf{U}_G)_{1:L^{\text{top}}}\mathbf{\Lambda}_{1:L^{\text{top}}}(\mathbf{U}_A \otimes \mathbf{U}_G)_{1:L^{\text{top}}}^T$ ,  $(\mathbf{U}_A \otimes \mathbf{U}_G)_{1:K^{\text{top}}}\mathbf{\Lambda}_{1:K^{\text{top}}}(\mathbf{U}_A \otimes \mathbf{U}_G)_{1:K^{\text{top}}}^T$  and  $(\mathbf{U}_A \otimes \mathbf{U}_G)\mathbf{\Lambda}_{1:L}(\mathbf{U}_A \otimes \mathbf{U}_G)^T = (\mathbf{U}_A \otimes \mathbf{U}_G)_{1:L}\mathbf{\Lambda}_{1:L}(\mathbf{U}_A \otimes \mathbf{U}_G)_{1:L}^T$  are again positive semi-definite by the fact that low rank approximation does not introduce negative eigenvalues.

Now, a well-known fact from dimensional reduction literature is that low rank approximation preserving the top eigenvalues results in best approximation errors in terms of Frobenius norm for the given rank. Informally stating Wely's ideas on eigenvalue perturbation:

Let  $\mathbf{B} \in \mathbb{R}^{m \times n}$  with rank smaller or equal to  $p$  (one can also use complex space  $\mathbb{C}$  instead of  $\mathbb{R}$ ) and let  $\mathbf{E} = \mathbf{A} - \mathbf{B}$  with  $\mathbf{A} \in \mathbb{R}^{m \times n}$ . Then, it follows that,

$$\begin{aligned} \|\mathbf{A} - \mathbf{B}\|_F^2 &= \Sigma_1(\mathbf{A} - \mathbf{B})^2 + \dots + \Sigma_\mu(\mathbf{A} - \mathbf{B})^2 \\ &\geq \Sigma_{p+1}(\mathbf{A} - \mathbf{B})^2 + \dots + \Sigma_\mu(\mathbf{A} - \mathbf{B})^2 \\ &= \|\mathbf{A} - \mathbf{B}_{1:p}\|_F^2, \end{aligned} \tag{B.20}$$

where  $\Sigma_1, \dots, \Sigma_\mu$  are the singular values of  $\mathbf{A}$  with  $\mu = \min(n, m)$ . The convention here is that  $\Sigma_i(\mathbf{A})$  is the  $i$ -th largest singular value and  $\Sigma_i(\mathbf{A}) = 0$  for  $i > \text{rank}(\mathbf{A})$ . Using this insight, and the fact that in the given settings, squared singular values are variances in new space leads to:

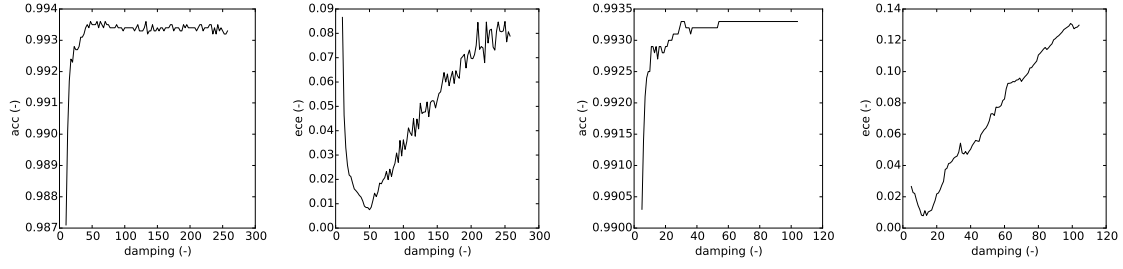
$$\|\mathbf{F} - \hat{\mathbf{F}}_{1:K}^{\text{top}}\|_F \geq \|\mathbf{F} - \hat{\mathbf{F}}_{1:L}\|_F \geq \|\mathbf{F} - \hat{\mathbf{F}}_{1:L}^{\text{top}}\|_F.$$

This completes the proof of the Lemma.  $\square$

## B.2. Implementation Details

The following experiments are implemented using Tensorflow (Abadi et al., 2016): (a) toy regression, (b) UCI benchmark in the next section, (c) active learning and (d) classification on MNIST and CIFAR10. The KFAC library from Tensorflow<sup>2</sup> was used to implement the Fisher estimator (Martens & Grosse, 2015) for our methods and the works of Ritter et al. (2018b). On the other hand, Pytorch (Paszke et al., 2019) has been used for ImageNet and adversarial defense experiments. The plug-and-play code is made available for Pytorch.

<sup>2</sup>Available at <https://github.com/tensorflow/kfac>



**Figure B.1:** Grid search results. For Diag (left two figures) and KFAC (right two) an extensive grid search has been conducted to ensure fair comparison. Here, we report the results with pseudo observation term of 50000 on MNIST. This ensures that a main difference to DEF Laplace is the expression for model uncertainty as the inference and network architectures are kept the same.

Note that empirical Fisher usually is not a good estimate of the Hessian, as it is typically biased (Martens & Grosse, 2015; Kunstner et al., 2019). Instead, KFAC library offers several estimation modes. We have used the gradients mode for KFAC whereas the exact mode was used for Diag. NVIDIA Tesla and 1080Ti are used for all the experiments.

### B.2.1. Implementation Details: Section 4.5.1

The training details are as follows: Adam has been used with a learning rate of 0.001 with zero prior precision or L2 regularization coefficient ( $\tau = 0.2$  for KFAC,  $\tau = 0.45$  for Diag,  $N = 1$  and  $\tau = 0$  for both EFB and INF have been used). Mean squared error (MSE) loss is used. The exact block-wise Hessian and their approximations for the given setup contained zero values on its diagonals. This can be interpreted as zero variance in the IM, meaning no information, resulting in an IM being degenerate for the likelihood term. In such cases, the covariance may not be uniquely defined (Thrun et al., 2004). Therefore, we treated these variances as deterministic, making the IM non-degenerate (similar findings reported by MacKay (1992b)). We have used Numpyro (Phan et al., 2019) for the implementations of HMC, with 50000 samples. For BBB, we have used an open-source implementation<sup>3</sup> where the Gaussian noise is sampled in a batch initially, and symmetric sampling is deployed. Lastly,  $T = 100$  samples were used.

### B.2.2. Implementation Details: Section 4.5.2

Details of experiment settings are as follows: we split each one into training, validation, and test set with 20, 100 (which is reasonable when compared with the size of test set) and 100 data points randomly for 20 times, respectively. The remaining points serve as pool set, in which we are not allowed to obtain their labels. So we have 20 splits in this experiment. Once the model chooses the data point in the pool set and moves it into the training set, its label becomes available, which simulates the scenario where humans annotate it. The experiments progress as follows: firstly, we trained the model with the initial training set and select one point from the pool set which will be put into the training set with its label. Then we train the model again and proceed into the next iteration. In each iteration, we evaluate our model on the test set and report the root mean square error (RMSE). We select the model during training and the corresponding hyperparameter ( $\tau$ ) based on its performance on the validation set. While the range of  $N$  is  $[0.5 * N, 1.0 * N]$ , where  $N$  is

<sup>3</sup>Available at <https://github.com/ThirstyScholar/bayes-by-backprop>

MNIST	Dim N [-]	Dim L [-]	Percent [%]
<i>CNN-1</i>	800	450	<b>56.25</b>
<i>CNN-2</i>	51200	5185	<b>10.12</b>
<i>FC-1</i>	3211264	5625	<b>0.18</b>
<i>FC-2</i>	10240	4775	<b>46.63</b>
CIFAR	Dim N [-]	Dim L [-]	Percent [%]
<i>CNN-1</i>	4800	4800	<b>100</b>
<i>CNN-2</i>	102400	2112	<b>2.06</b>
<i>FC-1</i>	884736	3980	<b>0.45</b>
<i>FC-2</i>	73728	5499	<b>7.45</b>
<i>FC-3</i>	1920	1920	<b>100</b>

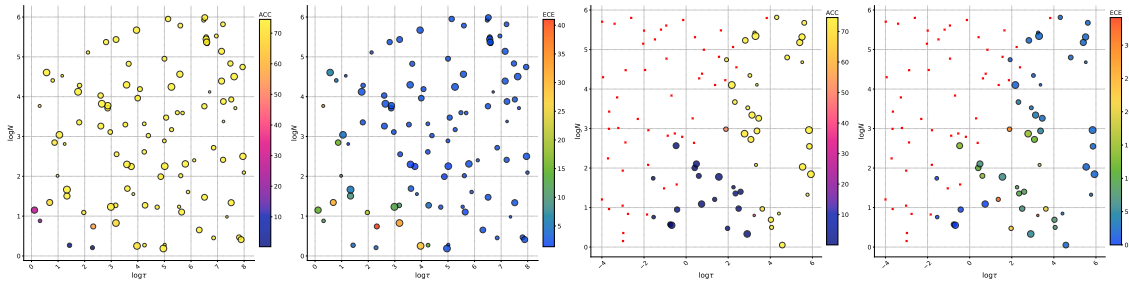
**Table B.2.:** Necessity of low rank approximation and reduction in complexity. Reduced dimensions from N to the chosen rank L per layer are reported for both MNIST and CIFAR10 experiments. CNN stand for convolution while FC is for fully connected layers. The complexity of sampling  $O(N^3)$  are reduced to  $O(L^3)$ . Employed strategy here was to keep the maximum for the rank K, which results in a seemingly arbitrary rank L.

the size of the training set in the current iteration. The range of  $\tau$  is  $[1, 200, 400]$ . For other hyperparameters, we use a learning rate of 0.01 for boston housing and energy, 0.001 for wine and L2 regularization of 0.0 for boston housing and wine,  $1e-5$  for energy. The mini-batch size is set to the initial size of the training set, 20. Only 1 point is selected in each iteration and the number of iterations is set to 20. Regarding model selection, we use early stopping based on the RMSE on the validation set and the maximum epoch is set to 40.

### B.2.3. Implementation Details: Section 4.5.3

For MNIST, the dropout layer is used in the FC layers with a rate of 0.6. An important piece of information is the size of each layer. The first layer constitutes 32 filters with a five by five kernel, followed by the second layer with 64 filters and a five by five kernel. The first fully connected layer then constitutes 1024 units and the last one ends with ten units. We note that this validates our method on memory efficiency, as the third layer has a large number of parameters, and its covariance, being quadratic in its size, cannot be stored in our utilized GPUs. For CIFAR10 experiments, the most relevant settings are: the first layer constitutes a five by five kernel with 64 filters. This is then again followed by the same. Units of 384, 192, and 10 have been used for the fully connected layers. Lastly, random cropping, flipping, brightness changes, and contrast are the applied data augmentations.

Implementation of deep ensemble ([Lakshminarayanan et al., 2017](#)) is kept rather simple by not using adversarial training, but we combined 15 networks that were trained with different initializations. The same architecture and training procedure were used for all. For dropout, we have tried a grid search of dropout probabilities of 0.5 and 0.8, and have reported the best results. For the methods based on LA, we have performed a grid search on hyperparameters N of (1, 50000, 100000) and 100 values of  $\tau$  were tried using a known class validation set. Note that for every method and different datasets, each method required different values of  $\tau I$  to give a reasonable accuracy. Figure B.1 depicts examples on MNIST where minimum ECE was selected. The LRA is imposed as a way to tackle the challenges of computational intractability. To empirically assess the reduction in complexity, we depict the parameter and low rank dimensions N and L respectively in table B.2. As shown, LRA-based sampling computations reduce the computational complexity significantly. Furthermore, this explains the necessity of LRA - certain layers (e.g. FC-1 of both MNIST



**Figure B.2.:** Hyperparameter search Results of random hyperparameter search for DenseNet121. From left to right: Diag Acc., Diag ECE, KFAC Acc. and KFAC ECE. Red crosses indicate configurations that were not invertible due to degeneracy or numerical instability. For accuracy, higher is better while for ECE lower is better.

and CIFAR experiments) are computationally intractable to store, infer, and sample.

#### B.2.4. Implementation Details: Section 4.5.4

To demonstrate that our method does not require changes in the training procedure, we used pre-trained weights from Pytorch<sup>4</sup>. Results are discussed in section B.3.6. SWAG and SWA are the available baselines which have been evaluated on the ImageNet dataset and implementations are officially open-sourced<sup>5</sup>. We closely followed the described experimental procedure. For the out-of-domain data, we have used artistic impressions and paintings of landscapes and objects<sup>6</sup>. Lastly, performing multiple forward passes for the entire validation set is still a computationally expensive task. We therefore chose  $T = 30$  during inference, which we empirically found sufficient for convergence, similar to Maddox et al. (2019).

We performed an extensive hyperparameter search for all LA methods using 100 randomly sampled pairs of  $N$  and  $\tau$  selected from a log-scale between zero and ten. We resorted to random search instead of grid search, as it tends to yield stronger results with a smaller number of samples (Bergstra & Bengio, 2012). The results for the accuracy (Acc.) and expected calibration error (ECE) are shown in figure B.2. The ECE can be extremely low in insufficiently regularized areas because the accuracy is also very low there, which is why we show the results for both metrics.

### B.3. Additional Experiments and Results

#### B.3.1. UCI Benchmarks

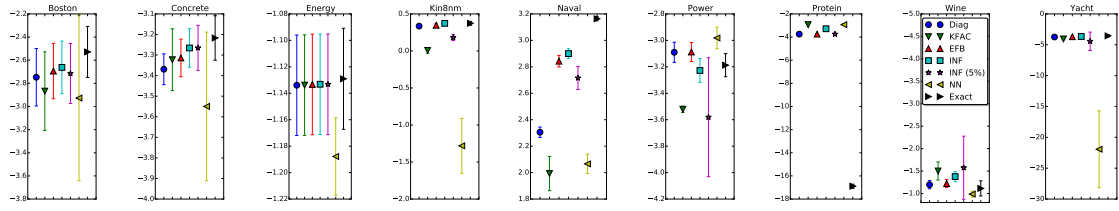
Experiments on UCI benchmark<sup>7</sup> have been conducted to evaluate various IM estimates and their effects on predictive uncertainty estimation. We evaluate LA-based approaches only, which have the advantage that the only differences between each approach are approximations of IM. To explain, inference principle, network architectures, and training convergence can be kept the same. We note that this is difficult for approaches based on variational inference. Due to this, meaningful comparisons can often be difficult, as specific

<sup>4</sup>Available at <https://pytorch.org/docs/stable/torchvision/models.html>

<sup>5</sup>Available at [https://github.com/wjmaddox/swa\\_gaussian](https://github.com/wjmaddox/swa_gaussian)

<sup>6</sup>Available at <https://www.kaggle.com/c/painter-by-numbers/data>

<sup>7</sup>Dataset and splits have been taken from <https://github.com/yaringal/DropoutUncertaintyExps>.



**Figure B.3.:** Evaluating predictive uncertainty on UCI datasets. We report test log likelihood on the y-axis and compare Diag, KFAC, EFB, INF variants, NN and Exact. Here, exact refers Laplace Approximation using the block-wise exact information matrix while NN denotes a deterministic neural network. Using linear approximation to the predictive uncertainty (MacKay, 1992b), accuracy between each methods are kept the same, which ensures fair comparisons using test log likelihood. Higher the better.

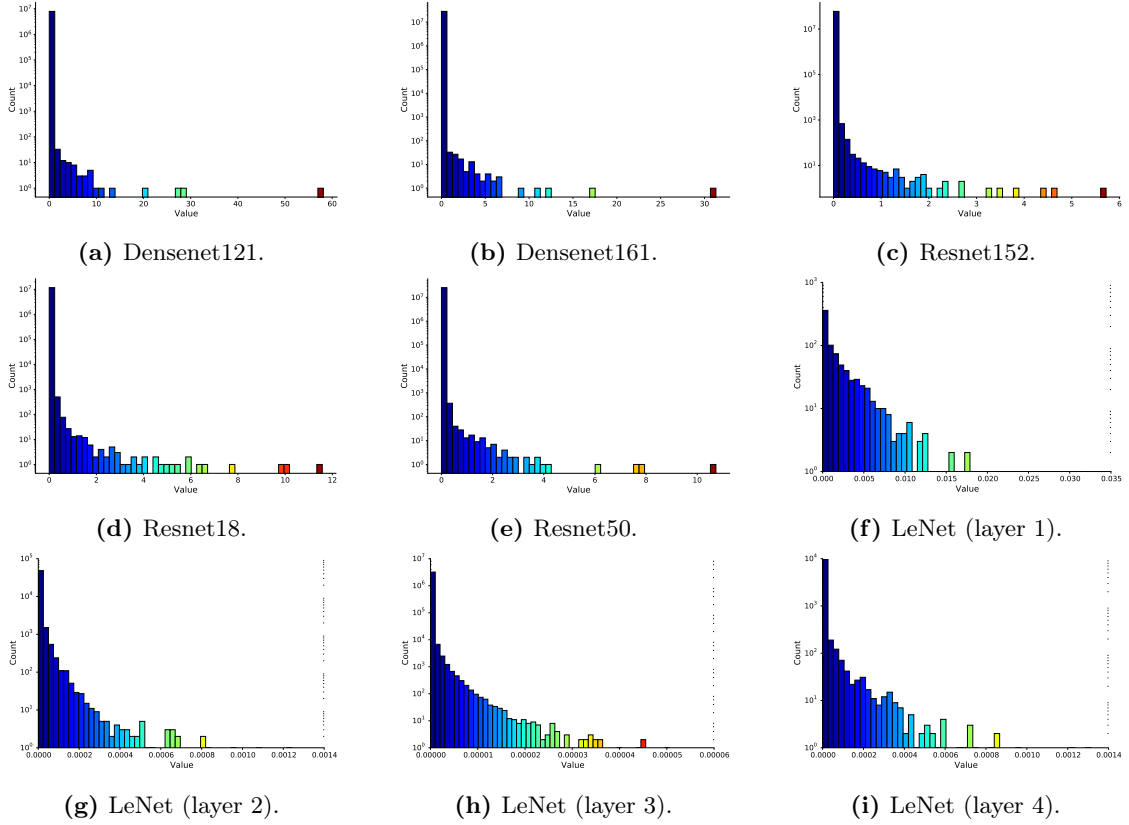
Datasets	Boston	Concrete	Energy	Kin8nm	Naval	Power	Protein	Wine	Yacht
RMSE	3.361±0.929	6.181±0.727	0.573±0.070	0.164±0.005	0.010±0.0001	4.322±0.153	4.516±0.123	0.637±0.034	9.568±1.132

**Table B.3.:** UCI benchmark. Root mean squared error (RMSE) is reported for the used set-up of UCI benchmark experiments. Note that as test log-likelihood depends on accuracy (in addition to uncertainty estimation), we have used linearized LA on the output space so that the test accuracy or RMSE is kept the same among the compared LA-based approaches. Our model overfit in some datasets so that effectiveness of having Bayesian Neural Network can be seen clearly. This also does not affect our results as all LA-based methods are built on top of the same model.

details such as the number of epochs can have significant effects on the results (Mukhoti et al., 2018). In this line of argument, we have further used the linearized LA (MacKay, 1992b; Foong et al., 2019) instead of sampling-based evaluation (Gal, 2016; Ritter et al., 2018b). Linearized LA is presented in chapter 2.

As shown, the mean of prediction depends only on the new test data. As the test log-likelihood depends on the accuracy of the predictor, we can keep the accuracy of the predictors the same among various LA-based approaches (reported in table B.3). Furthermore, as the covariance matrix for predictive uncertainty only depends on *aleatoric* uncertainty, gradients on the new test input and epistemic uncertainty, comparisons between LA-based approaches are simpler as the experiments can be implemented in a way that the difference lies only in various approximations of epistemic uncertainty. Closely following Foong et al. (2019), layer-wise exact IM has been established and the same hyperparameter settings are applied across other LA-based approaches<sup>8</sup>. Figure B.3 reports the results of UCI experiments where we compare the reliability of uncertainty estimates using the test log-likelihood as a measure. As shown, we find that our approach compares well to the others. Note that in Power and Protein, LA approaches were under-performing even when compared to a deterministic DNN. This is a known limitation of LA: the approximated posterior may cover areas of low probability mass, and the approaches perform similarly to a deterministic DNN or become unstable. Here, our experiments also indicate that improvements in terms of Frobenius norm of error may not directly translate to performance in uncertainty estimation, at least for LA, which requires in-depth treatment for future works.

<sup>8</sup>We also present the results of sampling based evaluations in section B.3.5 where we extensively search hyperparameters for each LA methods separately.



**Figure B.4.:** Eigenvalue histogram. For (a)-(e), the eigenvalues of ImageNET architectures are shown. Here, x-axis plots the values whereas y-axis shows the counts in a log scale. From figure (f)-(i), we show the eigenvalues of MNIST (layer-wise differentiated). These figures empirically shows that the spectrum of information matrix is sparse (tend to have many values close to zeros) for all the considered architectures. Furthermore, in the considered set-up for MNIST dataset, more over-parameterized layer tends to have more close-to-zero eigenvalues even within the same architecture.

### B.3.2. Spectral Sparsity of Information Matrix

One of the key insights behind our work is that the information matrix of over-parameterized DNNs tends to have close to zero eigenvalues (equivalently sparse in its spectrum). Is this true for the considered experiments? To answer this question, we plot the eigenvalue histograms in figure B.4. Figure B.4 shows that the empirical findings of [Sagun et al. \(2018\)](#) hold well in our experiment setup. Two concrete observations are found: (a) with varying depth (figures B.4a, B.4b, B.4c, B.4d, B.4e), IM showed a tendency to get more sparse (especially on the maximum eigenvalue), and (b) with varying numbers of parameters in each layer (figures B.4f, B.4g, B.4h, B.4i) IM showed to be more sparse with the number of parameters. One possible insight is that the information of individual parameters tends to be smaller if there are more parameters for explaining the same amount of data.

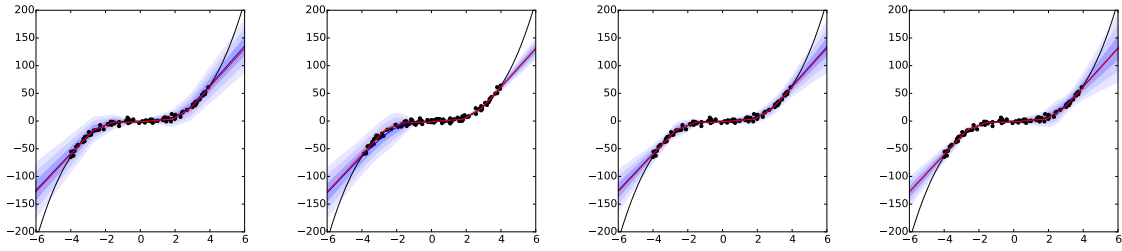
### B.3.3. Effects of Low Rank Approximation

We additionally study the effects of LRA on the approximation quality of IM when compared to the exact, block diagonal IM. We include exact diagonal approximation to the true IM while decreasing the ranks of INF in steps of 25%. The results are depicted in table



Dataset	Off-diagonals Diag	INF (75%)	INF (50%)	INF (25%)
<i>Boston</i>	$1.000 \pm 0.000$	$0.524 \pm 0.006$	$0.524 \pm 0.006$	$0.520 \pm 0.006$
<i>Concrete</i>	$1.000 \pm 0.000$	$0.506 \pm 0.008$	$0.506 \pm 0.008$	$0.508 \pm 0.008$
<i>Energy</i>	$1.000 \pm 0.000$	$0.504 \pm 0.006$	$0.504 \pm 0.006$	$0.514 \pm 0.006$
<i>Kin8nm</i>	$1.000 \pm 0.000$	$0.526 \pm 0.005$	$0.526 \pm 0.005$	$0.546 \pm 0.005$
<i>Naval</i>	$1.000 \pm 0.000$	$0.465 \pm 0.003$	$0.465 \pm 0.003$	$0.465 \pm 0.003$
<i>Power</i>	$1.000 \pm 0.000$	$0.492 \pm 0.008$	$0.492 \pm 0.008$	$0.502 \pm 0.008$
<i>Protein</i>	$1.000 \pm 0.000$	$0.541 \pm 0.021$	$0.541 \pm 0.021$	$0.541 \pm 0.021$
<i>Wine</i>	$1.000 \pm 0.000$	$0.535 \pm 0.009$	$0.535 \pm 0.009$	$0.546 \pm 0.009$
<i>Yacht</i>	$1.000 \pm 0.000$	$0.516 \pm 0.007$	$0.516 \pm 0.007$	$0.526 \pm 0.007$

**Table B.4.:** UCI benchmark: The normalized Frobenius norm of errors for the off-diagonal approximations w.r.t the true Fisher are depicted.



**Figure B.5.:** Uncertainty on toy regression. The black dots and the black lines are data points ( $x$ ,  $y$ ). The red and blue lines show predictions of the deterministic Neural Network and the mean output respectively. Up to three standard deviations are shown with blue shades.

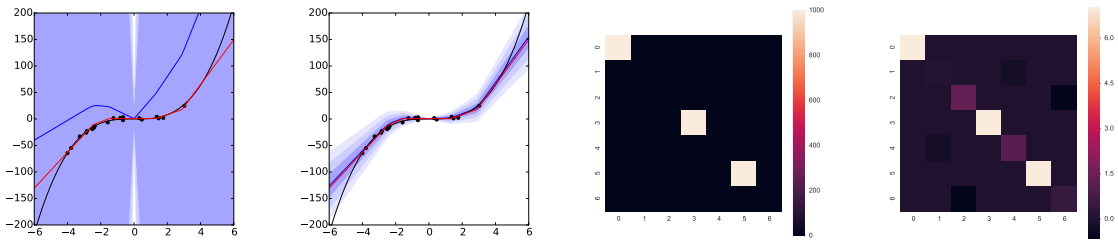
B.4 which shows that due to the sparsity of IM, the error (with a measure on normalized Frobenius norm) does not drastically increase with lower ranks. Diagonal approximation also results in the most severe approximation error on off-diagonal elements.

### B.3.4. Additional Results on Toy Regression

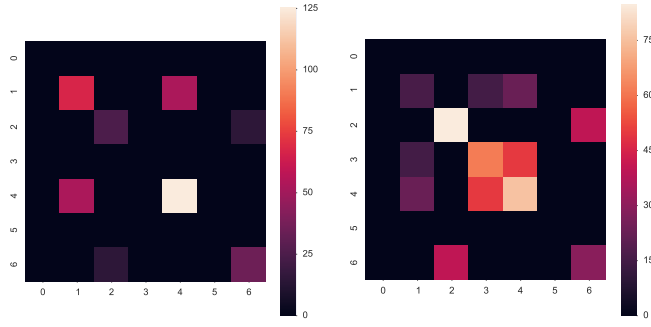
#### Effects of diagonal correction

What is the relation between keeping diagonal elements of IM exact and predictive uncertainty? As the effects of regularizing hyperparameters are removed to a certain extent in the toy experiment, we study the above-mentioned question within this limited but controllable set-up.

For this purpose, we depict Diag, EFB, FB (layer-wise true IM) and INF with rank one. The most comparable fit to HMC is given by FB, while INF with rank one deteriorates



**Figure B.6.:** Toy regression uncertainty and covariance visualization (only the first layer is shown here). OKF Laplace means using equation B.21 without further approximation in equation B.22.



**Figure B.7:** Visualization of the approximate information matrix with different data points. Only the first layer chosen for the analysis. With increasing data points, the resulting information matrix becomes less degenerate.

when compared to its full-rank counterpart. EFB for this set-up produces considerable misfit to HMC. Importantly, since the only difference between EFB and DEF Laplace is a diagonal correction term, these results suggest that keeping diagonals of IM exact can result in accurate predictive uncertainty.

**KFAC - a critical analysis.** Ritter et al. (2018b,a) reports that KFAC requires smaller sets of hyperparameters than Diag, which may suggest that KFAC produces better fits to the true posterior. Instead, we find that KFAC’s approximation step for the prior incorporation may result in this phenomenon. Concretely, let’s define two variants as:

$$\mathbf{N}\mathbf{F}_{\text{kfac}} + \tau\mathbf{I} = \mathcal{N}(\mathbf{A} \otimes \mathbf{G}) + \tau\mathbf{I} \quad \text{or} \quad (\text{B.21})$$

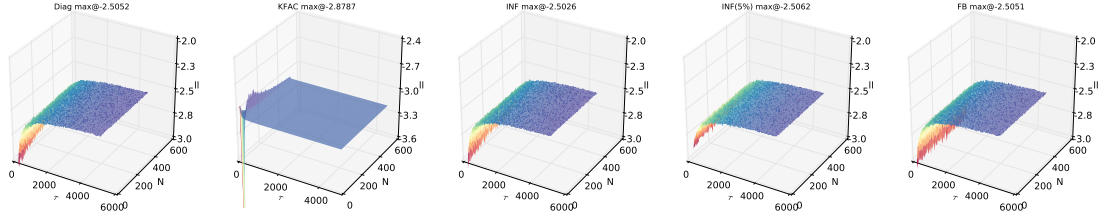
$$\mathbf{N}\mathbf{F}_{\text{kfac}} + \tau\mathbf{I} \approx \left(\sqrt{\mathbf{N}}\mathbf{A}_{i-1} + \sqrt{\tau}\mathbf{I}\right) \otimes \left(\sqrt{\mathbf{N}}\mathbf{G}_i + \sqrt{\tau}\mathbf{I}\right). \quad (\text{B.22})$$

Here, equation B.22 has been the approximation step of Ritter et al. (2018b,a) while equation B.21 is an exact variant. We denote the latter as OKF. By reproducing the results of Ritter et al. (2018b), we depict the results in figure B.6, in which we plot the predictive uncertainty obtained from OKF and KFAC under the same hyperparameter settings. Furthermore, a direct plot of the covariance matrix can be found as well for the same hyperparameters. These results show that without the approximation step in equation B.22, KFAC requires higher regularization hyperparameters, similar to Diag. Looking into the covariance matrix directly, we also find that the magnitude of KFAC is smaller with this approximation. Therefore, our findings are that KFAC, due to the given approximation in the incorporation of prior, requires smaller sets of regularization hyperparameters. Furthermore, as OKF does not seem to result in a similar phenomenon, it might be difficult to conclude that KFAC, when compared to Diag, produces a better fit to the true posterior.

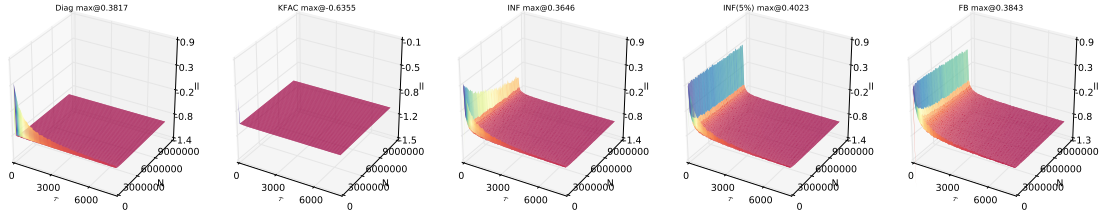
**Effects of data points size.** We now study the effects of dataset size to number of parameters. For this, we compare the dataset sizes 100 and 20. Results are depicted in figure B.7. Notably, using 20 data points resulted in a greater number of zero diagonal entries and corresponding rows and columns. This might be due to the over-parameterization of the model, which results in an underdetermined Hessian.

### B.3.5. Effects of Hyperparameters - UCI benchmark

Instead of linearized LA, we investigate the performance of LA-based methods with a full Bayesian analysis on UCI benchmarks. Rather than reporting the best performance of each method with a single selected hyperparameter choice, we perform extensive grid searches and show the performance landscape. Such performance landscape can be informative for studying how a more accurate approximation of the information matrix translates to



**Figure B.8.:** Boston Hyperparameter Landscape. Test-log likelihood on the z-axis. Ranging hyperparameters are display on XY plane. Maximum values are also displayed for Diag, KFAC, INF with two different ranks, and FB (true block-diagonal information matrix). Except KFAC, all LA-based approaches show similar behavior. Concrete, energy and protein showed similar tendency.



**Figure B.9.:** Kin8nm Hyperparameter Landscape. Test-log likelihood on the z-axis. Ranging hyperparameters are display on XY plane. Maximum values are also displayed for Diag, KFAC, INF with two different ranks, and FB (true block-diagonal information matrix). All LA-based approaches show different behavior. Naval, power, wine and yacht datasets have similar tendency.

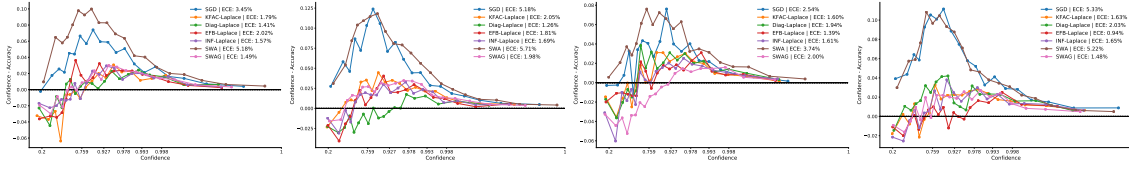
uncertainty estimation under the effects of hyperparameters. Note that this is possible on UCI datasets due to the small scale of the setup.

To this end, we search 10,000 hyperparameter sets for each method except KFAC, where we increase the size to 20,000 hyperparameters<sup>9</sup>. The range of hyperparameter sets has been chosen differently for each dataset so that all the methods produce reliable predictions. We draw 100  $K_{mc}$  samples for each prediction in order to have an acceptable range of convergence for Monte Carlo integration. Results are shown in figures B.8 and B.9 where we report the following observations that fall into two categories.

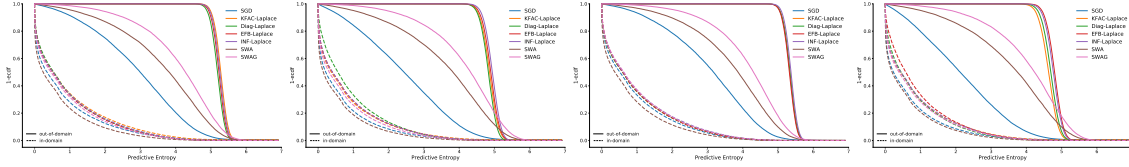
- **Type 1 landscape:** Experiments on datasets namely boston housing, concrete, energy and protein showed similar behaviors. As seen in figure B.8, the performance landscape (test log-likelihood) show similar curves for all the methods except KFAC. The maximum achievable performance have been found also similar with a marginal difference.
- **Type 2 landscape:** Experiments on datasets namely kin8nm, naval, power, wine and yacht showed a different tendency than type 1. While no methods significantly outperformed the other uniformly across all the datasets, the curve showed different behavior than type 1 as reported in figure B.9. Each methods also showed different performance landscape.

These experiments suggest that for type 1, the benefits from having more accurate Fisher information are marginal. This suggests that improvements in the accuracy of IM w.r.t. Frobenius norm of error may not directly translate to more accurate uncertainty estimation within this context of LA.

<sup>9</sup>We have doubled the search space for KFAC as it requires smaller sets of hyperparameters due to equation B.22 instead of equation B.21. Following this observation, we further decreased the minimum  $\tau$ .



**Figure B.10.:** Calibration results on large scale experiments: From left to right: ResNet50, ResNet152, DenseNet121 and DenseNet161. Our method tends to outperform SWA and SWAG while being competitive to other fine tuned LA-based approaches.



**Figure B.11.:** Out-of-domain: From left to right: ResNet50, ResNet152, DenseNet121 and DenseNet161. Our method tends to significantly outperform SWA and SWAG while being competitive to other fine tuned LA-based approaches.

For type 2 however, interesting differences can be found in the sense that INF variants and FB showed significantly more regimes of hyperparameter sets that output higher log-likelihood, which can be benefits of having a more accurate Fisher information matrix - when only a smaller number of hyperparameter searches are possible, a more accurate IM can result in better quality of predictive uncertainty. Understanding the causes of these behaviors to full generality seems a challenging research question as LA is tightly coupled with the loss landscape of DNNs and further, how optimization affects generality and the shape of the true posterior. One possible explanation for type 1 is that maintaining a single  $\tau$  and  $N$  for all the layers may force all the methods to be regularized for fitting a few sharply peaked local modes of the true posterior, hindering the benefits of having more accurate estimates of the true Fisher information.

### B.3.6. Additional ImageNet Results

The calibration and OOD detection experiments presented in the main text on ResNet18 were performed identically for the four additional architectures. We show the results in figures B.10 and B.11. The observations from the main text hold for the additional networks. All LA-based approaches can reduce the calibration error significantly compared to the deterministic network and SWA and are as good as or better than SWAG. In out-of-domain separation, we find that the LA-based approaches perform comparably strongly and are far superior to the other methods across all considered networks.

Architecture	Diag [ms]	KFAC [ms]	EFB [ms]	INF [ms]
<i>ResNet18</i>	$1.24 \pm 0.06$	$8.23 \pm 0.04$	$9.28 \pm 0.06$	$4.74 \pm 0.08$
<i>ResNet50</i>	$1.94 \pm 0.11$	$15.47 \pm 0.18$	$16.89 \pm 0.09$	$12 \pm 0.25$
<i>ResNet152</i>	$5.4 \pm 0.07$	$32.08 \pm 0.5$	$35.55 \pm 0.07$	$32.62 \pm 0.15$
<i>DenseNet121</i>	$4.41 \pm 0.14$	$8.92 \pm 0.19$	$10.22 \pm 0.13$	$25.03 \pm 0.65$
<i>DenseNet161</i>	$5.86 \pm 0.11$	$16.48 \pm 0.03$	$18.84 \pm 0.54$	$35.95 \pm 0.35$

**Table B.5.:** Wall clock time analysis on sampling. Mean and standard deviation over 1000 draws are reported with a single thread.

Table B.5 also the wall clock analysis for sampling. Interestingly, for ResNet variants, INF is more efficient than KFAC and EFB due to the effects of low rank approximation. On the other hand, DenseNet variants have many small layers and therefore, rank reduction is less noticeable and cannot outweigh the disadvantage of having a larger number of smaller operations in a sampling procedure. While KFAC and EFB maintain similar size matrices, EFB sampling is slower than KFAC, also due to a larger number of operations. Diag is, as expected, the most efficient method. We note, however, that Bayesian Neural Networks in general have a disadvantage that prediction time is at least 30 times slower (assuming 30 samples are taken) and thus, there may not be any practical advantages.

Architecture	Diag [min]	KFAC [min]	EFB [min]	INF [min]
<i>ResNet18</i>	30	120	165	165
<i>ResNet50</i>	82	210	300	300
<i>ResNet152</i>	180	510	720	720
<i>DenseNet121</i>	100	360	465	465
<i>DenseNet161</i>	180	870	1060	1060

**Table B.6.: Wall clock time analysis on information matrix computation.** Values are rounded to the nearest.

Next, table B.6 reports the wall clock analysis for IM computations. In our implementation, EFB is computed after having KFAC and therefore, it takes more time than KFAC. The original implementation of EFB contains amortized eigen-decomposition and can be made more efficient than KFAC. INF is an offline procedure and provides negligible overhead to EFB. The total computation time for all the methods is less than a day on ImageNet, and thus, this analysis shows the practicality and scalability of LA-based approaches.

## C.1. Derivations and Proofs

### C.1.1. Derivations of Neural Networks and Local GPs Connections

In chapter 5, we have outlined our main idea, i.e. projecting the NLMs to function-space in order to obtain local GPs, which divide the input space into smaller local regimes, where the individual GP experts learn and make predictions. In this section, we describe the theoretical contributions of our work, which focus on establishing the mathematical relationships between DNNs and local GPs.

Before explaining our contributions, we stress the relevance of the theoretical work within the context of the work. First, as we derive our theory in this section, several insights behind our method are revealed. These include various design choices that motivate the individual components of our algorithm. Second, the theory provides a foundation to the related algorithms. In practice, we argue that the proposed practical algorithm is only one way of exploiting the derived theory. We hope that such theoretic foundations can help the community to build on our work for developing more effective uncertainty quantification techniques. Lastly, several works have established the relationships between DNNs and GPs (Neal, 1996a; Khan et al., 2019; Lee et al., 2018a; Jacot et al., 2018). In this sense, our work extends the prior work towards better understandings of DNNs.

#### Mixtures of Neural Network Experts

Consider again a supervised learning task on a dataset consisting of input-output pairs  $\mathcal{D} = \{\mathcal{X}, \mathcal{Y}\} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N$ , where  $\mathbf{x}_i \in \mathbb{R}^D$ ,  $\mathbf{y}_i \in \mathbb{R}^K$ . Whenever possible, we drop the indices  $i$  for simplicity. Defining a neural network as a  $\boldsymbol{\theta} \in \mathbb{R}^P$  parametrized function  $f_{\boldsymbol{\theta}} : \mathbb{R}^D \rightarrow \mathbb{R}^K$  that maps the inputs  $\mathcal{X}$  to the outputs  $\mathcal{Y}$ , learning seeks to obtain an empirical risk minimizer of the loss function, i.e.  $\min_{\boldsymbol{\theta}} \frac{1}{|\mathcal{D}|} \sum_{(\mathbf{x}, \mathbf{y}) \in \mathcal{D}} \mathcal{L}(f_{\boldsymbol{\theta}}(\mathbf{x}), \mathbf{y}) + \frac{\delta}{2} \boldsymbol{\theta}^T \boldsymbol{\theta}$  where  $\delta$  is an  $L_2$  regularizer. Here, a mini-batch  $\mathcal{B} \subset \mathcal{D}$  is often used instead to find a local maximum-a-posteriori (MAP) solution  $\hat{\boldsymbol{\theta}}$ . Importantly, we assume a twice differentiable and strictly convex loss function  $\mathcal{L}$  and piece-wise linear activations in  $f_{\boldsymbol{\theta}}$ . For example, this includes square loss or cross-entropy loss with RELU, which are often used in modern

DNNs.<sup>1</sup>

Now, a Mixture of Expert (MoE) consists of  $M$  experts defined as learners  $\mathcal{F} = \{f_{\theta_1}, \dots, f_{\theta_M}\}$ , and a gating function  $g: \mathbb{R}^D \rightarrow \Delta^{M-1}$  that maps any input  $\mathbf{x}$  to  $g(\mathbf{x}) = [g_1(\mathbf{x}), \dots, g_M(\mathbf{x})]$  (Jacobs et al., 1991). Each expert of the MoE learns and predicts within a subset of the input domain, and a gating function generates these subsets. We depict an example on the right side of figure 5.2, where the experts are GPs. Here, the gating function divides the input space  $\mathcal{X}$  and assigns each datum  $\mathbf{x}$  to an individual GP expert. We denote such models as local GPs (or MoE-GP) (Tresp, 2001; Rasmussen & Ghahramani, 2002), and further also define a MoE-DNN, where each of the experts are DNNs (Blum et al., 2018; Mees et al., 2016) (refer to the left side of figure 5.2). Importantly, for MoE-DNNs, we assume a gating function:  $g_m(\mathbf{x}) = 1$  in just one coordinate for each input (Gross et al., 2017) where the subscripts  $m = 1, 2, \dots, M$  denote the  $m^{\text{th}}$  expert. For the partitioned data  $\mathcal{D} = \{\mathcal{D}_1, \mathcal{D}_2, \dots, \mathcal{D}_M\}$  the training minimizes an individual loss function:  $\min_{\theta_m} \frac{1}{|\mathcal{D}_m|} \sum_{(\mathbf{x}, \mathbf{y}) \in \mathcal{D}_m} \mathcal{L}(f_{\theta_m}(\mathbf{x}), \mathbf{y}) + \frac{\delta}{2} \theta_m^T \theta_m$ . For a clear exposition, we distinct the notation of ground truth  $\mathbf{y}$  with  $\mathbf{y} = \sum_{m=1}^M g_m(\mathbf{x}) f_{\theta_m}(\mathbf{x})$  to represent the prediction in the remaining texts.

### Laplace Approximation for Individual Experts

Next, consider a Bayesian treatment of MoE-DNNs with a priori regard to the parameters of individual DNN experts  $p(\theta_m)$ , and also their posterior distributions given the data  $p(\theta_m | \mathcal{D}_m) \forall m$ . Concretely, we employ the Laplace Approximation (MacKay, 1992c) to compute the posterior probabilities of the DNN experts, which can be obtained by taking the second-order Taylor series expansion of the log posterior. Using a Gaussian prior with precision  $\delta_m$ , we obtain:

$$\log p(\theta_m | \mathcal{D}_m) \approx \log p(\hat{\theta}_m | \mathcal{D}_m) + \frac{1}{2}(\theta_m - \hat{\theta}_m)^T (\mathbf{H}_m + \delta_m \mathbf{I})(\theta_m - \hat{\theta}_m),$$

where the first-order term vanishes as the gradient of the log posterior  $\nabla \log p(\theta_m | \mathcal{D}_m)$  is close to zero at  $\hat{\theta}_m$ . Taking the exponential on both sides and approximating integrals by reverse engineering densities, the weight posterior is approximately a Gaussian with mean  $\hat{\theta}_m$  and covariance matrix  $\Sigma_m = (\mathbf{H}_m + \delta_m \mathbf{I})^{-1}$  where  $\mathbf{H}_m$  is the Hessian of  $\log p(\theta_m | \mathcal{D}_m)$  (MacKay, 1992c; Bishop & Nasrabadi, 2006).

Unfortunately, working with  $\mathbf{H}_m$  is difficult, as it is not in general positive semi-definite (PSD). Fortunately, when using standard loss and piece-wise linear activations, a good approximation of  $\mathbf{H}_m$  is given by Gauss-Newton Matrix (Martens & Grosse, 2015):

$$\mathbf{H}_m \approx \frac{1}{|\mathcal{D}_m|} \sum_{(\mathbf{x}, \mathbf{y}) \in \mathcal{D}_m} \mathbf{J}_{f_m}(\mathbf{x})^T \mathbf{H}_{\mathcal{L}_m}(\mathbf{x}, \mathbf{y}) \mathbf{J}_{f_m}(\mathbf{x}), \quad \text{where}$$

$$\mathbf{J}_{f_m}(\mathbf{x}) = \frac{\partial f_{\theta_m}(\mathbf{x})}{\partial \theta_m^T}, \quad \mathbf{H}_{\mathcal{L}_m}(\mathbf{x}, \mathbf{y}) = \frac{\partial^2 \mathcal{L}(f_{\theta_m}(\mathbf{x}), \mathbf{y})}{\partial f_{\theta_m}(\mathbf{x})^T \partial f_{\theta_m}(\mathbf{x})}, \quad \mathbf{R}_{\mathcal{L}_m}(\mathbf{x}, \mathbf{y}) = \frac{\partial \mathcal{L}(f_{\theta_m}(\mathbf{x}), \mathbf{y})}{\partial f_{\theta_m}(\mathbf{x})}, \quad (\text{C.1})$$

are the Jacobian of  $f_{\theta_m}(\mathbf{x})$  wrt. parameters  $\theta_m$ , the Hessian of the loss function  $\mathcal{L}(\mathbf{x}, \mathbf{y})$  wrt.  $f_{\theta_m}(\mathbf{x})$ , and a residual vector  $\mathbf{R}_{\mathcal{L}_m} \in \mathbb{R}^K$  respectively. Notice that  $\mathbf{H}_{\mathcal{L}_m}(\mathbf{x}, \mathbf{y}) \in \mathbb{R}^{K \times K}$ ,  $\mathbf{J}_{f_m}(\mathbf{x}) \in \mathbb{R}^{K \times P}$  and hence  $\mathbf{H}_m \in \mathbb{R}^{P \times P}$ . The result of these steps turns a mixture of DNN experts into a mixture of Bayesian Neural Network experts, where the parameters are represented using Gaussian distributions.

<sup>1</sup>Our theories are for any DNN architecture that satisfies the given assumption. Only feedforward network shown in figure 5.2, but our approach also applies to convolution, and recurrent neural networks



### Neural Networks as Mixtures of Gaussian Process Experts

So far, we have defined MoE-DNNs with a hard gating function, and then showed how their parameters' posterior can be approximated with a Gaussian rather than the point estimates  $\hat{\theta}_m$  for all  $m$ . This step results in a MoE model, where the experts are Bayesian Neural Networks. Having these essentials, we make the following statement (a specific instance of [Khan et al. \(2019\)](#), which is in turn based on the NLMs ([MacKay, 1992b](#))).

**Proposition C.1.1. (Local Duality)** *Let the posterior for each Deep Neural Network expert  $p(\theta_m|\mathcal{D}_m)$  be approximated with a Gaussian distribution:  $p(\theta_m|\mathcal{D}_m) \sim \mathcal{N}(\hat{\theta}_m, \Sigma_m)$ . Define a transformed dataset  $\tilde{\mathcal{D}}_m = \{\mathcal{X}_m, \tilde{\mathcal{Y}}_m\}$  with pseudo output datum  $\tilde{\mathcal{Y}}_m = \{\tilde{\mathbf{y}}_{m,1}, \dots, \tilde{\mathbf{y}}_{m,N_m}\}$  such that  $\tilde{\mathbf{y}}_{m,j_m} := \mathbf{J}_{f_m}(\mathbf{x})\hat{\theta}_m - \mathbf{H}_{\mathcal{L}_m}(\mathbf{x}, \mathbf{y})^{-1}\mathbf{R}_{\mathcal{L}_m}(\mathbf{x}, \mathbf{y})$  for  $j_m = 1, 2, \dots, N_m$ . Then,  $\tilde{\mathbf{y}}_m = \mathbf{J}_{f_m}(\mathbf{x})\theta_m + \epsilon_m$  where  $\epsilon_m \sim \mathcal{N}(\mathbf{0}, \mathbf{H}_{\mathcal{L}_m}(\mathbf{x}, \mathbf{y})^{-1})$  and  $\theta \sim \mathcal{N}(\mathbf{0}, \delta_m^{-1}\mathbf{I})$  is a neural linear model that has an equivalent posterior  $p(\theta_m|\tilde{\mathcal{D}}_m) \sim \mathcal{N}(\hat{\theta}_m, \Sigma_m) \forall m$ .*

*Proof.* The proof can be found in the appendix C.1.3. □

*Remark C.1.2.* We remark that for each DNN expert with Laplace Approximation, there is a counterpart linear model with feature maps  $\mathbf{J}_{f_m}(\mathbf{x})$  that has the same posterior distribution as the original DNN expert. This is visualized in figure 5.2 in a function-space view, mapping a linear model to a GP.

An alternative path to obtain the NLMs is a linearization trick of [MacKay \(1992b\)](#). [MacKay \(1992b\)](#) applies the first-order Taylor series to the output of the neural network. Then, considering the Bayesian formulation of linear models with Gaussian priors and the same Gaussian posterior distributions of the original network (obtained using Laplace Approximation), [MacKay \(1992b\)](#) shows that the predictive uncertainty can be obtained in a closed-form solution, due to the conjugating properties of Gaussians for linear models. An elegant work of [Khan et al. \(2019\)](#) extends [MacKay \(1992b\)](#) by pointing out that we can recover GPs from a neural network with Gaussian posterior distributions. Our proof here is a specific instance of these two works - we project the same set-up within MoE models with a hard gating function.

Now, we extend to consider MoEs with a hard gating function globally. Having the individual DNN experts having equivalent GPs (as stated in the previous proposition), we show similar derivation steps on how MoEs with DNNs have an equivalent posterior distribution with local GPs.

**Lemma C.1.3. (Global Duality)** *Let  $m$  Deep Neural Network experts form a hard mixture of experts model such that  $\mathbf{y} = \sum_{m=1}^M g_m(\mathbf{x})f_{\theta_m}(\mathbf{x})$ . We denote its Laplace Approximation-based posterior as  $p(\theta|\mathcal{D})$ . Under a transformation  $\tilde{\mathcal{D}} = \{\tilde{\mathcal{D}}_1, \dots, \tilde{\mathcal{D}}_M\}$ ,  $\tilde{\mathbf{y}} = \sum_{m=1}^M g_m(\mathbf{x})\tilde{f}_{m_\theta}(\mathbf{x})$  is a mixture of experts with a linear regressor that has an equivalent posterior distribution  $p(\theta|\tilde{\mathcal{D}})$ .*

*Proof.* The proof can be found in appendix C.1.3. □

*Remark C.1.4.* This Lemma states that if we assume a hard MoE where the experts are DNNs with Gaussian posteriors (obtained using Laplace Approximation), there then exists a hard MoE with the NLMs, which is dictated by the same gating function and has equal distribution of the parameters given the data. This establishes the duality of the two models in a Bayesian sense, where the models are represented by the same probability distribution. Intuitively, such relationships can be obtained as we take a hard gating function, making each DNN expert probabilistically independent.

An important ramification of this result can be obtained in a well-known equivalence of weight-space and function-space view. To explain, we can obtain stochastic processes:

$$\tilde{\mathbf{y}} = \sum_{m=1}^M g_m(\mathbf{x}) \tilde{f}_{\text{GP}_m}(\mathbf{x}) + \epsilon \quad \text{with} \quad (\text{C.2})$$

$$\tilde{f}_{\text{GP}_m}(\mathbf{x}) \sim \text{GP}(\mathbf{0}, \frac{1}{\delta_m} \mathbf{J}_{f_m}(\mathbf{x})^T \mathbf{J}_{f_m}(\mathbf{x})), \quad (\text{C.3})$$

which is a local GP with NTK, i.e. a tangent kernel with  $\mathbf{J}_{f_m}(\mathbf{x})$ . Consequently, the generative model and the predictions  $\mathcal{N}(\tilde{\mathbf{y}}_m^*, \tilde{\Sigma}_m(\mathbf{x}^*))$  on the new test datum  $\mathbf{x}^*$  are given by:

$$\begin{aligned} \begin{bmatrix} \tilde{\mathbf{y}}_m \\ \tilde{f}_{\text{GP}_m}(\mathbf{x}^*) \end{bmatrix} &\sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K}_m(\mathcal{X}, \mathcal{X}) + \sigma_{0,m} \mathbf{I} & \mathbf{k}_m(\mathcal{X}, \mathbf{x}^*) \\ \mathbf{k}_m(\mathbf{x}^*, \mathcal{X}) & \mathbf{k}_m(\mathbf{x}^*, \mathbf{x}^*) \end{bmatrix}\right) \\ \tilde{\mathbf{y}}_m^*(\mathbf{x}^*) &= \mathbf{k}_{m,*}^T (\mathbf{K}_m + \sigma_{0,m} \mathbf{I})^{-1} \tilde{\mathbf{y}}_m, \\ \tilde{\Sigma}_m(\mathbf{x}^*) &= \mathbf{k}_m(\mathbf{x}^*, \mathbf{x}^*) - \mathbf{k}_{m,*}^T (\mathbf{K}_m + \sigma_{0,m} \mathbf{I})^{-1} \mathbf{k}_{m,*} + \sigma_{0,m}, \end{aligned} \quad (\text{C.4})$$

respectively. Here,  $\mathbf{K}_m(\mathcal{X}, \mathcal{X}) = \mathbf{K}_m = \frac{1}{\delta_m} \mathbf{J}_{f_m}(\mathbf{x})^T \mathbf{J}_{f_m}(\mathbf{x})$ , and  $\mathbf{k}_m(\mathcal{X}, \mathbf{x}^*) = \mathbf{k}_{m,*}$ . Also, data uncertainty is captured by  $\sigma_{0,m} \mathbf{I}$ . Overall, with these steps, we show the duality of MoEs with DNNs and local GPs in a Bayesian sense. We note that if MoEs with DNNs and Gaussian approximations exist, we can exploit their equivalent local GPs without any approximation errors. Unfortunately, as MoEs with DNNs may often not be used in practice, we expand our theoretic work to a single DNN.

Therefore, we now establish a connection between a DNN and local GP as shown in figure 5.2, in order to increase the applicability of our theoretical results. To do so, we point out that a single trained DNN can be treated as a MoE-DNN if all the DNN experts are essentially the same DNNs and a hard gating function is designed to strictly divide the input space among the experts, i.e. only one local DNN expert per division.

**Proposition C.1.5. (Equivalence in Input-Output Relationships)** *Let  $f_{\theta}$  be a deterministic deep neural network with the input-prediction set given by  $\{(\mathbf{x}_i, f_{\theta}(\mathbf{x}_i))\}_{i=1}^N$  for all  $N$  data points. Define a mixture of experts model, with a hard gating function  $g_m(\mathbf{x}_i)$  and the neural network experts  $f_{\theta_m} = f_{\theta}$  for all  $m = 1, \dots, M$  experts. Then, the mixture of experts model has an equivalent input-prediction set to the deterministic deep neural network, given by  $\{(\mathbf{x}_i, f_{\theta}(\mathbf{x}_i))\}_{i=1}^N$ .*

*Proof.* The proof can be found in appendix C.1.4. □

**Remark C.1.6.** Under these given conditions, the input-prediction relationships of the above two models are the same for both train and test data. This implies that (a) a single DNN can also be seen as a hard MoE with GPs, as illustrated in figure 5.2, and (b) we assume that the data is stationary. We note that obtaining a hard MoE with DNNs, under the stated conditions, may not necessarily involve a training step.

One can imagine training a single DNN on the entire dataset, and theoretically, this is the same as having a hard MoE with the same trained DNNs for the predictions within the training and test set. For example, let's try to form an ensemble of DNNs with exactly the same network with respect to the structure of the learner and the network parameters. Moreover, imagine a predictive model where we do not average the ensemble of DNNs for

predictions, but pick only a single member of the ensemble per input data. Intuitively, the input-prediction relationships are the same as a single DNN with the same network parameters and structure.

What does then this proposition imply to the GP experts? So far, we have (a) shown the equivalence between MoEs with DNNs and GPs previously, and (b) established a connection between MoEs with DNNs to a single DNN. This is by introducing the specific conditions on the gating function, and how MoEs with DNNs are formed. Considering the GP experts, the given extension from MoEs with DNNs to a single DNN implies  $\mathbf{K}_1 = \mathbf{K}_2 = \dots = \mathbf{K}_M$  for all  $M$ . This is because we have taken a single DNN as the DNN experts within the MoEs model and brought an assumption that the data is stationary. We note that the overall kernel matrix can model non-stationary processes as it is still possible to keep different hyperparameters  $\delta_m$  and  $\sigma_m$ . We present a formal statement about the proposed approximation, when compared to a single DNN with Gaussian approximations, which has a true equivalent GPs  $f_{\text{GP}}$ .

**Lemma C.1.7. (On Approximation Error)** *Let  $f_{\text{GP}}(\mathbf{x})$  be a true stationary process such that  $\mathbf{K}_m(\cdot, \cdot) = \mathbf{K}(\cdot, \cdot) \forall m$ . Then, our approximation to  $\mathbf{K}_{\text{true}}(\mathcal{X}, \mathcal{X})$  is  $\mathbf{K}(\mathbf{x}_i, \mathbf{x}_j) = \mathbb{1}_{\mathbf{c}_i = \mathbf{c}_j} \mathbf{K}(\mathbf{x}_i, \mathbf{x}_j)$  where  $\mathbf{c}_i$  is the expert assignment for  $\mathbf{x}_i$ , indicator function  $\mathbb{1}_{a=b} = 1$  when  $a = b$  and 0 otherwise. Then, the approximation error of the local GP kernel is  $\|\mathbf{K}(\mathcal{X}, \mathcal{X}) - \mathbf{K}_{\text{true}}(\mathcal{X}, \mathcal{X})\|_F^2 = \sum_{ij} \mathbf{K}(\mathbf{x}_i, \mathbf{x}_j)^2 - \sum_{m=1}^M \sum_{ij \in \mathcal{Y}_m} \mathbf{K}(\mathbf{x}_i, \mathbf{x}_j)^2$ .*

*Proof.* The proof can be found in appendix C.1.4. □

**Remark C.1.8.** Here,  $\sum_{ij} \mathbf{K}(\mathbf{x}_i, \mathbf{x}_j)^2$  is a constant while the term  $\sum_{m=1}^M \sum_{ij \in \mathcal{Y}_m} \mathbf{K}(\mathbf{x}_i, \mathbf{x}_j)^2$  specifies that less approximation error occurs when less correlated data points (or off-diagonal elements) are assumed to be independent while strongly correlated points by NTK should be within the same GP experts. In other words, the more block diagonal a true equivalent GP, the better local GPs can approximate it.

Intuitively, a MoE-DNN with Gaussian approximations and a local GP have the equivalent posterior distribution, where GPs have their kernel matrix defined by their counterpart DNN experts. If we force a MoE-DNN to a single DNN with the stated conditions, the GP experts have the kernel matrix, which is defined by the counterpart DNN experts that have the same posterior distribution. This Lemma quantifies the result of this step, and as the GP experts have the same kernel function, the given local GP assumes stationary data.

We discussed the derivations and the results of our theoretic work, where a relationship has been established between a DNN and local GPs with NTK. In summary, we showed how a MoE-DNN and local GPs are dual in a Bayesian sense, and further established a connection to a single DNN. With this, we have gained insight into the proposed idea, i.e. the consequences of the resulting approximations and the conditions on when the two models are equivalent. We note that the gained insight has been explored in the design of our gating function. Next, we provide more theoretic justifications, and the key benefits that stem from the derived theory.

### Theoretic Justifications and Key Benefits

Our proposed predictive model comprises a DNN for accurate predictions, as well as a local GP with NTK for predictive uncertainty. We first justify the resulting predictive model as follows.

Consider the square loss  $\mathcal{L}_m = \frac{1}{2} (\mathbf{y}_m - f_{\boldsymbol{\theta}_m}(\mathbf{x}_m))^2$  so that  $\mathbf{R}_{\mathcal{L}_m}(\mathbf{x}) = \sigma_m^{-2} (f_{\boldsymbol{\theta}_m}(\mathbf{x}) - \mathbf{y}_m)$  and  $\mathbf{H}_{\mathcal{L}_m}(\mathbf{x}) = \sigma_m^{-2} \mathbf{I}$ , the mapping between  $\mathcal{D}_m$  and  $\tilde{\mathcal{D}}_m$  are simply related by  $\tilde{\mathbf{y}}_m^* =$

$\mathbf{J}_{f_m}(\mathbf{x}^*)\hat{\boldsymbol{\theta}}_m - (f_{\boldsymbol{\theta}_m}(\mathbf{x}^*) - \mathbf{y}^*)$  for new predictions  $\mathbf{y}^*$  and  $\tilde{\mathbf{y}}_m^*$ . Rearranging the terms simply gives us:  $\mathbf{y}^* = \tilde{\mathbf{y}}_m^* + f_{\boldsymbol{\theta}_m}(\mathbf{x}^*) - \mathbf{J}_{f_m}(\mathbf{x}^*)\hat{\boldsymbol{\theta}}_m$ . Now, the variance will be the same for both  $\mathbf{y}^*$  and  $\tilde{\mathbf{y}}_m^*$  due to other terms being constant deterministic variables. This justifies the use of a GP model for uncertainty estimates, on top of DNN predictions. We show several more examples below from our experiments (section C.1.2). Note that the outputs of our GPs are transformed outputs  $\tilde{\mathbf{y}}_m^*$ , which are parametrized by the DNNs via  $\mathbf{J}_{f_m}(\mathbf{x}^*)$  and  $\hat{\boldsymbol{\theta}}_m$ . Thus, we can keep the DNN predictions  $f_{\boldsymbol{\theta}}(\mathbf{x})$  and disregard the mean outputs of the GPs.

It is crucial to note that there are several benefits of our proposed predictive model. First, by the design, we can inherit the benefits of the mixture of GP experts model (Tresp, 2001; Rasmussen & Ghahramani, 2002) for uncertainty estimates. These are (a) a divide-and-conquer principle that significantly reduces the computational complexity of the GPs, (b) the ability to model non-stationary covariance for both epistemic and aleatoric uncertainty, and (c) a natural support for distributed computation. Second, the prediction accuracy is the same as for the original DNN while the uncertainty estimates can be computed from the equivalent GP model. In other words, we attain the best of both worlds: the predictive power of DNNs and the uncertainty estimates from scalable GPs. This is achieved in a decoupled manner, e.g. the DNN does not require re-training, nor degrade its predictive performance. And finally, as shown in equation 5.4, our method does not require multiple forward passes through the DNNs. This is in contrast to many existing methods which often require multiple samples of the DNN predictions for a single input datum (either from a deep ensemble (Lakshminarayanan et al., 2017) or model uncertainty (Gal & Ghahramani, 2016)). Hence, the resulting probabilistic predictor is fast and accurate, which is of importance to robotics.

### C.1.2. Derivations of Covariances per Loss Functions

Our framework uses the loss functions in order to derive the relationships between the predictions of DNNs and GPs. This means that per loss function, we need a derivation for the model information (e.g. the first or second order information such as Hessian or residuals). Here, we provide a derivation for two loss functions that are used in the considered network architectures of our experiments. These include mean squared errors and cross entropy. Unlike the rest of the chapter, we omit superscript  $m$ , that indicates a specific expert, but the discussion herein applies to any  $m$ . This is for the clarity of the exposition.

#### Inverse Dynamics Task

For the inverse dynamics task, we used a multi-layer perceptron (MLP) with mean squared error as the loss function  $\mathcal{L}$  per single input. It is defined as:

$$\mathcal{L} = \frac{1}{2} (\mathbf{y} - f_{\boldsymbol{\theta}}(\mathbf{x}))^2, \quad (\text{C.5})$$

where the  $\mathbf{x}$  is the input point,  $\mathbf{y}$  is the measured output (7 joints),  $f_{\boldsymbol{\theta}}(\mathbf{x})$  is the predictions of the trained MLP. Then, assuming an independent GP per joint, the Hessian  $\mathbf{H}_{\mathcal{L}_m} = 1$ , the residual  $\mathbf{R}_{\mathcal{L}_m} = (f_{\boldsymbol{\theta}}(\mathbf{x}) - \mathbf{y})$ , and consequently, the transformed output  $\tilde{\mathbf{y}} = \mathbf{J}_f(\mathbf{x})\hat{\boldsymbol{\theta}} + f_{\boldsymbol{\theta}}(\mathbf{x}) - \mathbf{y}$ . Here, the quantities we infer are  $f_{\boldsymbol{\theta}}(\mathbf{x})$  and  $\tilde{\mathbf{y}}$ , and for predictive uncertainty:  $\text{Cov}(\tilde{\mathbf{y}}) = \text{Cov}(\mathbf{J}_f(\mathbf{x})\hat{\boldsymbol{\theta}} + f_{\boldsymbol{\theta}}(\mathbf{x}) - \mathbf{y}) = \text{Cov}(f_{\boldsymbol{\theta}}(\mathbf{x}))$ . This theoretically justifies that GP, trained on  $(\mathbf{x}, \tilde{\mathbf{y}})$ , can be used for uncertainty estimates, while keeping the original MLP prediction  $f_{\boldsymbol{\theta}}(\mathbf{x})$ .

## Object Detection Task

Next, we describe the mathematical derivation related to the DETR architecture for object detection. DETR is a state-of-the-art object detection method that uses transformers and is also built on an end-to-end principle. Object detection involves both classification and regression (bounding box), which extends the previously described cases that only considered a regression task. Let  $\mathbf{y}$  be the ground truth with  $\mathbf{y}_i = (\mathbf{c}_i, \mathbf{b}_i)$ . Here,  $\mathbf{c}_i$  is the class labels, and  $\mathbf{b}_i \in [0 \ 1]^4$  is the representation of a bounding box with centers, height, and width of the box. DETR uses  $K$  object queries (decoded by the transformer layer), followed by MLPs that overall predict  $\mathbf{y} = \{\hat{\mathbf{y}}_i\}_{i=1}^K$ . To match the number  $K$ , the ground truth label  $\mathbf{y}$  is padded with  $\emptyset$  (no object) into  $K$ .

For training, DETR first performs matching between the supervised labels and the predictions (each representing a set):

$$\hat{\sigma} = \arg \min_{\sigma \in \emptyset} \sum_i^K \mathcal{L}_{\text{match}}(\mathbf{y}_i, \mathbf{y}_{\sigma(i)}) \quad \text{with} \quad (C.6)$$

$$\mathcal{L}_{\text{match}}(\mathbf{y}_i, \mathbf{y}_{\sigma(i)}) = -\mathbf{1}_{\{\mathbf{c}_i \neq \emptyset\}} \hat{\mathbf{p}}_{\hat{\sigma}(i)}(\mathbf{c}_i) + \mathbf{1}_{\mathcal{L}_{\text{box}}}(\mathbf{b}_i, \hat{\mathbf{b}}_{\hat{\sigma}(i)}).$$

The match index is denoted  $\sigma(i)$ . Given the optimal match with respect to the above criteria, the used loss is defined:

$$\mathcal{L}_{\text{DETR}} = \sum_i^K \left[ -\log \hat{\mathbf{p}}_{\hat{\sigma}(i)}(\mathbf{c}_i) + \mathbf{1}_{\{\mathbf{c}_i \neq \emptyset\}} \|\mathbf{b}_i, \hat{\mathbf{b}}_{\hat{\sigma}(i)}\|_2^2 \right]. \quad (C.7)$$

The above loss is back-propagated whenever the object query receives the optimal assignment  $\hat{\sigma}$  (equivalently, equation C.6 is solved using the Hungarian algorithm, and is not back-propagated).

Now, notice that per input datum  $\mathbf{x}$ , the regression loss, whenever  $\mathbf{c}_i \neq \emptyset$ , is simply the MSE loss, and otherwise, does not exist. This means that our derivation only needs to take into account the bounding box uncertainty when  $\mathbf{c}_i \neq \emptyset$ . Then, using the same techniques as before, the transformed bounding box output is:  $\tilde{\mathbf{b}}_{\hat{\sigma}(i)} = \mathbf{J}_f(\mathbf{x}_k) \hat{\boldsymbol{\theta}} - \hat{\mathbf{b}}_{\hat{\sigma}(i)} + \mathbf{b}_i$  for all  $i$ , and  $\text{Cov}(\tilde{\mathbf{b}}_{\hat{\sigma}(i)}) = \text{Cov}(\mathbf{J}_f(\mathbf{x}_k) \hat{\boldsymbol{\theta}} - \hat{\mathbf{b}}_{\hat{\sigma}(i)} + \mathbf{b}_i) = \text{Cov}(\hat{\mathbf{b}}_{\hat{\sigma}(i)})$ . This justifies the use of GP uncertainty for bounding box regression.

For classification loss  $-\log \hat{\mathbf{p}}_{\hat{\sigma}(i)}(\mathbf{c}_i)$ , which is the broadly used cross-entropy loss, we now derive the relationship between the transformed output and true output. Let  $\mathbf{c}_i$  be given by  $h[f_{i,\boldsymbol{\theta}}(\mathbf{x})]$ , where  $h$  is the sigmoid function (used in DETR), and  $f_{i,\boldsymbol{\theta}}(\mathbf{x})$  is the activation of the classification layer for input point  $\mathbf{x}$ . It implies that DETR uses the Bernoulli likelihood function, and we may treat the classification as regression (similar to [Lu et al. \(2020\)](#)). Then, the residual term is given by  $h[f_{i,\boldsymbol{\theta}}(\mathbf{x})] - \mathbf{c}_i$  and the Hessian of the sigmoid is:  $h[f_{i,\boldsymbol{\theta}}(\mathbf{x})](1 - h[f_{i,\boldsymbol{\theta}}(\mathbf{x})])$ . Similar to a regression task, the transformed output:

$$\tilde{\mathbf{c}}_i = \mathbf{J}_f(\mathbf{x}) \hat{\boldsymbol{\theta}} - [h[f_{i,\boldsymbol{\theta}}(\mathbf{x})](1 - h[f_{i,\boldsymbol{\theta}}(\mathbf{x})])]^{-1} (h[f_{i,\boldsymbol{\theta}}(\mathbf{x})] - \mathbf{c}_i) \quad \text{or} \quad (C.8)$$

$$\mathbf{c}_i = h(f_{i,\boldsymbol{\theta}}(\mathbf{x})) + [h[f_{i,\boldsymbol{\theta}}(\mathbf{x})](1 - h[f_{i,\boldsymbol{\theta}}(\mathbf{x})])] \tilde{\mathbf{c}}_i - [h[f_{i,\boldsymbol{\theta}}(\mathbf{x})](1 - h[f_{i,\boldsymbol{\theta}}(\mathbf{x})])] \mathbf{J}_f(\mathbf{x}) \hat{\boldsymbol{\theta}}. \quad (C.9)$$

Lastly, taking the variance on both sides:

$$\begin{aligned}
\text{Cov}(\mathbf{c}_i) &= \text{Cov}\left(h(f_{i,\theta}(\mathbf{x}))\right) \\
&\quad + \left[h[f_{i,\theta}(\mathbf{x})](1 - h[f_{i,\theta}(\mathbf{x})])\right] \tilde{\mathbf{c}}_i - \left[h[f_{i,\theta}(\mathbf{x})](1 - h[f_{i,\theta}(\mathbf{x})])\right] \mathbf{J}_f(\mathbf{x}) \hat{\boldsymbol{\theta}}, \\
&= \left[h[f_{i,\theta}(\mathbf{x})](1 - h[f_{i,\theta}(\mathbf{x})])\right]^2 \text{Cov}(\tilde{\mathbf{c}}_i).
\end{aligned} \tag{C.10}$$

In the above equation, we reason about the variance of the class assignment. Overall, this derivation motivates the use of GP for classification uncertainty within DETR, as the variance of the transformed output is related to the original output. Then, we compute the Jacobian of the DNN for the given test input. This involves backpropagation of the DNNs and is again bounded to how efficient the DNN is. Then, our gating function assigns the test data to an expert, and we compute GP uncertainty.

### C.1.3. Proof of Lemma C.1.3

**Proposition C.1.1. (Local Duality)** *Let the posterior for each Deep Neural Network expert  $p(\boldsymbol{\theta}_m|\mathcal{D}_m)$  be approximated with a Gaussian distribution:  $p(\boldsymbol{\theta}_m|\mathcal{D}_m) \sim \mathcal{N}(\hat{\boldsymbol{\theta}}_m, \boldsymbol{\Sigma}_m)$ . Define a transformed dataset  $\tilde{\mathcal{D}}_m = \{\mathcal{X}_m, \tilde{\mathcal{Y}}_m\}$  with pseudo output datum  $\tilde{\mathcal{Y}}_m = \{\tilde{\mathbf{y}}_{m,1}, \tilde{\mathbf{y}}_{m,2}, \dots, \tilde{\mathbf{y}}_{m,N_m}\}$  such that  $\tilde{\mathbf{y}}_{m,j_m} := \mathbf{J}_{f_m}(\mathbf{x})\hat{\boldsymbol{\theta}}_m - \mathbf{H}_{\mathcal{L}_m}(\mathbf{x}, \mathbf{y})^{-1} \mathbf{R}_{\mathcal{L}_m}(\mathbf{x}, \mathbf{y})$  for  $j_m = 1, 2, \dots, N_m$ . Then,  $\tilde{\mathbf{y}}_m = \mathbf{J}_{f_m}(\mathbf{x})\boldsymbol{\theta}_m + \epsilon_m$  where  $\epsilon_m \sim \mathcal{N}(\mathbf{0}, \mathbf{H}_{\mathcal{L}_m}(\mathbf{x}, \mathbf{y})^{-1})$  and  $\boldsymbol{\theta} \sim \mathcal{N}(\mathbf{0}, \delta_m^{-1} \mathbf{I})$  is a neural linear model that has an equivalent posterior  $p(\boldsymbol{\theta}_m|\tilde{\mathcal{D}}_m) \sim \mathcal{N}(\hat{\boldsymbol{\theta}}_m, \boldsymbol{\Sigma}_m) \forall m$ .*

*Proof.* The proof sketch is as follows. To show the local equivalence between the Laplace approximated DNN experts posterior and that of a neural linear model, we first reformulate the DNN experts posterior  $p(\boldsymbol{\theta}_m|\mathcal{D}_m)$  with the so-called information form or natural parameters of Multivariate Normal Distribution (MND). Then, we demonstrate that the posterior of the individual  $m^{\text{th}}$  linear model can also be reformulated to match the given DNN experts posterior.

As stated, we reformulate the generalized Gauss Newton (GGN) approximated posterior distribution for individual experts as:

$$\begin{aligned}
p(\boldsymbol{\theta}_m|\mathcal{D}_m) &\sim \mathcal{N}(\hat{\boldsymbol{\theta}}_m, \boldsymbol{\Sigma}_m) \\
&= \frac{1}{\sqrt{(2\pi)^p |\boldsymbol{\Sigma}_m|}} \exp\left(-\frac{1}{2}(\boldsymbol{\theta}_m - \hat{\boldsymbol{\theta}}_m)^T \boldsymbol{\Sigma}_m^{-1}(\boldsymbol{\theta}_m - \hat{\boldsymbol{\theta}}_m)\right), \\
&= \exp\left(\mathbf{a} + \boldsymbol{\eta}^T \boldsymbol{\theta}_m - \frac{1}{2} \boldsymbol{\theta}_m^T \boldsymbol{\Lambda} \boldsymbol{\theta}_m\right).
\end{aligned} \tag{C.11}$$

where  $\mathbf{a}$  is the normalizing constant. In this formulation, MND is parameterized in the canonical form with the information vector  $\boldsymbol{\eta} = \boldsymbol{\Sigma}_m^{-1} \hat{\boldsymbol{\theta}}_m$  and matrix  $\boldsymbol{\Lambda} = \boldsymbol{\Sigma}_m^{-1}$ , instead of the mean and the covariance matrix  $(\hat{\boldsymbol{\theta}}_m, \boldsymbol{\Sigma}_m)$ . In the following, for better readability, we denote  $\mathbf{J}_{f_m}(\mathbf{x}_{m,i})$ ,  $\mathbf{R}_{\mathcal{L}_m}(\mathbf{x}_{m,i}, \mathbf{y}_{m,i})$ ,  $\mathbf{H}_{\mathcal{L}_m}(\mathbf{x}_{m,i}, \mathbf{y}_{m,i})$  by  $\mathbf{J}_{f_m,i}$ ,  $\mathbf{R}_{\mathcal{L}_m,i}$ ,  $\mathbf{H}_{\mathcal{L}_m,i}$ , respectively. The information vector and matrix can be expressed with the gradients and the Hessian:

$$\boldsymbol{\Sigma}_m^{-1} = \mathbf{H} + \delta_m \mathbf{I} \approx \sum_{i=1}^{N_m} \mathbf{J}_{f_m,i}^T \mathbf{H}_{\mathcal{L}_m,i} \mathbf{J}_{f_m,i} + \delta_m \mathbf{I}. \tag{C.12}$$

$$\begin{aligned}
\boldsymbol{\Sigma}_m^{-1} \hat{\boldsymbol{\theta}}_m &= -(\mathbf{J}_{f_m}^T \mathbf{R}_{\mathcal{L}_m} + \delta_m \hat{\boldsymbol{\theta}}_m) + \boldsymbol{\Sigma}_m^{-1} \hat{\boldsymbol{\theta}}_m, \\
&= \sum_{i=1}^{N_m} \left[ -\mathbf{J}_{f_m,i}^T \mathbf{R}_{\mathcal{L}_m,i} + \mathbf{J}_{f_m,i}^T \mathbf{H}_{\mathcal{L}_m,i} \mathbf{J}_{f_m,i} \hat{\boldsymbol{\theta}}_m \right],
\end{aligned} \tag{C.13}$$

We used the stationary assumption in Laplace Approximation, i.e. the loss is close to zero ( $-(\mathbf{J}_{f_m}^T \mathbf{R}_{\mathcal{L}_m} + \delta_m \hat{\boldsymbol{\theta}}_m) \approx 0$ ), and add  $\boldsymbol{\Sigma}_m^{-1} \hat{\boldsymbol{\theta}}_m$  on both sides. Again, we assume a Gaussian prior for the network parameters. Then, substituting this formulation of the information vector and matrix (equations C.13, C.12) into the given posterior distribution (equation C.11) results in:

$$\begin{aligned}
p(\boldsymbol{\theta}_m | \mathcal{D}_m) &\propto \exp\left(-\frac{1}{2} \boldsymbol{\theta}_m^T \left[ \sum_{i=1}^{N_m} \mathbf{J}_{f_m,i}^T \mathbf{H}_{\mathcal{L}_m,i} \mathbf{J}_{f_m,i} + \delta_m \mathbf{I} \right] \boldsymbol{\theta}_m \right. \\
&\quad \left. + \boldsymbol{\theta}_m^T \sum_{i=1}^{N_m} \left[ -\mathbf{J}_{f_m,i}^T \mathbf{R}_{\mathcal{L}_m,i} + \mathbf{J}_{f_m,i}^T \mathbf{H}_{\mathcal{L}_m,i} \mathbf{J}_{f_m,i} \hat{\boldsymbol{\theta}}_m \right] \right), \\
&\propto \exp\left(-\frac{\delta_m}{2} \boldsymbol{\theta}_m^T \boldsymbol{\theta}_m\right) \\
&\quad \prod_{i=1}^{N_m} \exp\left(-\frac{1}{2} \boldsymbol{\theta}_m^T \mathbf{J}_{f_m,i}^T \mathbf{H}_{\mathcal{L}_m,i} \mathbf{J}_{f_m,i} \boldsymbol{\theta}_m + \boldsymbol{\theta}_m^T \mathbf{J}_{f_m,i}^T \left[ \mathbf{H}_{\mathcal{L}_m,i} \mathbf{J}_{f_m,i} \hat{\boldsymbol{\theta}}_m - \mathbf{R}_{\mathcal{L}_m,i} \right] \right).
\end{aligned} \tag{C.14}$$

Now, we show that the posterior distribution of the neural linear model can also be written as equation C.14, which would complete the proof. Expressing the posterior  $p(\boldsymbol{\theta}_m | \tilde{\mathcal{D}})$ :

$$\begin{aligned}
&\propto \mathcal{N}(\boldsymbol{\theta}_m | 0, \delta_m^{-1} \mathbf{I}) \mathcal{N}(\tilde{\mathbf{y}}_m | \mathbf{J}_{f_m} \boldsymbol{\theta}_m, \mathbf{H}_{\mathcal{L}_m}^{-1}), \\
&\propto \exp\left(-\frac{\delta_m}{2} \boldsymbol{\theta}_m^T \boldsymbol{\theta}_m\right) \\
&\quad \prod_{i=1}^{N_m} \exp\left(-\frac{1}{2} (\tilde{\mathbf{y}}_{m,i} - \mathbf{J}_{f_m,i} \boldsymbol{\theta}_m)^T \mathbf{H}_{\mathcal{L}_m,i} (\tilde{\mathbf{y}}_{m,i} - \mathbf{J}_{f_m,i} \boldsymbol{\theta}_m)\right), \\
&\propto \exp\left(-\frac{\delta_m}{2} \boldsymbol{\theta}_m^T \boldsymbol{\theta}_m\right) \\
&\quad \prod_{i=1}^{N_m} \exp\left(-\frac{1}{2} \tilde{\mathbf{y}}_{m,i}^T \mathbf{H}_{\mathcal{L}_m,i} \tilde{\mathbf{y}}_{m,i} + \boldsymbol{\theta}_m^T \mathbf{J}_{f_m,i}^T \mathbf{H}_{\mathcal{L}_m,i} \tilde{\mathbf{y}}_{m,i} - \frac{1}{2} \boldsymbol{\theta}_m^T \mathbf{J}_{f_m,i}^T \mathbf{H}_{\mathcal{L}_m,i} \mathbf{J}_{f_m,i} \boldsymbol{\theta}_m\right).
\end{aligned} \tag{C.15}$$

Now, we can substitute  $\tilde{\mathbf{y}}_{m,i} := \mathbf{J}_{f_m,i} \hat{\boldsymbol{\theta}}_m - \mathbf{H}_{\mathcal{L}_m,i}^{-1} \mathbf{R}_{\mathcal{L}_m,i}$  into equation C.15. We note that:

$$-\frac{1}{2} \tilde{\mathbf{y}}_{m,i}^T \mathbf{H}_{\mathcal{L}_m,i} \tilde{\mathbf{y}}_{m,i} = -\frac{1}{2} [\mathbf{J}_{f_m,i} \hat{\boldsymbol{\theta}}_m - \mathbf{H}_{\mathcal{L}_m,i}^{-1} \mathbf{R}_{\mathcal{L}_m,i}]^T [\mathbf{H}_{\mathcal{L}_m,i} \mathbf{J}_{f_m,i} \hat{\boldsymbol{\theta}}_m - \mathbf{R}_{\mathcal{L}_m,i}], \tag{C.16}$$

is constant with respect to the random variables  $\boldsymbol{\theta}_m$ . Furthermore, we also obtain:

$$\begin{aligned}
\boldsymbol{\theta}_m^T \mathbf{J}_{f_m,i}^T \mathbf{H}_{\mathcal{L}_m,i} \tilde{\mathbf{y}}_{m,i} &= \boldsymbol{\theta}_m^T \mathbf{J}_{f_m,i}^T \mathbf{H}_{\mathcal{L}_m,i} [\mathbf{J}_{f_m,i} \hat{\boldsymbol{\theta}}_m - \mathbf{H}_{\mathcal{L}_m,i}^{-1} \mathbf{R}_{\mathcal{L}_m,i}] \\
&= \boldsymbol{\theta}_m^T \mathbf{J}_{f_m,i}^T [\mathbf{H}_{\mathcal{L}_m,i} \mathbf{J}_{f_m,i} \hat{\boldsymbol{\theta}}_m - \mathbf{R}_{\mathcal{L}_m,i}].
\end{aligned} \tag{C.17}$$

Substituting equations C.16 and C.17 into equation C.15 results in  $p(\boldsymbol{\theta}_m | \tilde{\mathcal{D}})$ :

$$\begin{aligned}
&\propto \exp\left(-\frac{\delta_m}{2} \boldsymbol{\theta}_m^T \boldsymbol{\theta}_m\right) \\
&\quad \prod_{i=1}^{N_m} \exp\left(-\frac{1}{2} \boldsymbol{\theta}_m^T \mathbf{J}_{f_m,i}^T \mathbf{H}_{\mathcal{L}_m,i} \mathbf{J}_{f_m,i} \boldsymbol{\theta}_m + \boldsymbol{\theta}_m^T \mathbf{J}_{f_m,i}^T [\mathbf{H}_{\mathcal{L}_m,i} \mathbf{J}_{f_m,i} \hat{\boldsymbol{\theta}}_m - \mathbf{R}_{\mathcal{L}_m,i}]\right) + \text{constant}.
\end{aligned} \tag{C.18}$$



This completes the proof.  $\square$

**Lemma C.1.3. (Global Duality)** *Let  $m$  Deep Neural Network experts form a hard mixture of experts model such that  $\mathbf{y} = \sum_{m=1}^M g_m(\mathbf{x}) f_{\theta_m}(\mathbf{x})$ . We denote its Laplace Approximation-based posterior as  $p(\theta|\mathcal{D})$ . Under a transformation  $\tilde{\mathcal{D}} = \{\tilde{\mathcal{D}}_1, \dots, \tilde{\mathcal{D}}_m\}$ ,  $\tilde{\mathbf{y}} = \sum_{m=1}^M g_m(\mathbf{x}) \tilde{f}_{m\theta}(\mathbf{x})$  is a mixture of experts with a linear regressor that has an equivalent posterior distribution  $p(\theta|\tilde{\mathcal{D}})$ .*

*Proof.* The proof sketch is as follows. We first show that for both  $\mathbf{y}$  and  $\tilde{\mathbf{y}}$ , the posterior distribution can be factorized as a product of individual  $m^{\text{th}}$  expert's posterior distribution:  $p(\theta_m|\mathcal{D}_m)$  and  $p(\theta_m|\tilde{\mathcal{D}})$ . Then, using the results of the previous Lemma, we prove that for a hard mixture of experts, the posterior distribution of two models is equivalent.

First, we express the likelihood of the MoE with DNNs as,

$$\begin{aligned} p(\mathcal{Y}|\mathcal{X}, \theta_m) &= \prod_i^N \sum_{m=1}^M g_m(\mathbf{x}_i) p(\mathbf{y}_i|\mathbf{x}_i, \theta_m), \\ &= \sum_{m=1}^M g_m(\mathbf{x}_1) p(\mathbf{y}_1|\mathbf{x}_1, \theta_m) \sum_{m=1}^M g_m(\mathbf{x}_2) p(\mathbf{y}_2|\mathbf{x}_2, \theta_m) \cdots \sum_{m=1}^M g_m(\mathbf{x}_N) p(\mathbf{y}_N|\mathbf{x}_N, \theta_m). \end{aligned} \quad (\text{C.19})$$

Here, we have assumed i.i.d data. Without loss of generality, we assumed an ordered assignment of data to individual experts, that is,  $i \in \{1, 2, \dots, N\}$  is decomposed into  $i \in \{i_1, i_2, \dots, i_M\}$  where for all  $m$ ,  $i_m \in \{1, 2, \dots, N^m\}$ . This means that each  $m^{\text{th}}$  expert is responsible for data points  $i_m$  such that  $g_m(\mathbf{x}_{i_m}) = 1$  for all  $i_m = 1, 2, \dots, N_m$ . Otherwise, the gating network outputs zero by definition. As a result, we can further decompose equation C.19:

$$\begin{aligned} &= \prod_{i_1}^{N_1} \sum_{m=1}^M g_m(\mathbf{x}_{i_1}) p(\mathbf{y}_{i_1}|\mathbf{x}_{i_1}, \theta_m) \prod_{i_2}^{N_2} \sum_{m=1}^M g_m(\mathbf{x}_{i_2}) p(\mathbf{y}_{i_2}|\mathbf{x}_{i_2}, \theta_m) \\ &\quad \cdots \prod_{i_M}^{N_M} \sum_{m=1}^M g_m(\mathbf{x}_{i_M}) p(\mathbf{y}_{i_M}|\mathbf{x}_{i_M}, \theta_m), \\ &= \prod_{i_1}^{N_1} p(\mathbf{y}_{i_1}|\mathbf{x}_{i_1}, \theta_m) \prod_{i_2}^{N_2} p(\mathbf{y}_{i_2}|\mathbf{x}_{i_2}, \theta_m) \cdots \prod_{i_M}^{N_M} p(\mathbf{y}_{i_M}|\mathbf{x}_{i_M}, \theta_m) \\ &= \prod_m^M \prod_{i_m}^{N_m} p(\mathbf{y}_{i_m}|\mathbf{x}_{i_m}, \theta_m). \end{aligned} \quad (\text{C.20})$$

We now define the prior over the parameters  $\theta$  by adopting the classical idea of automatic relevance determination (Neal, 1996a), which assumes a factorized Gaussian prior:

$$p(\theta) = \prod_{m=1}^M \mathcal{N}(\theta_m|\mathbf{0}, \mathbf{A}_m) \quad \text{where } \mathbf{A}_m = \text{diag}(\delta_1, \delta_2, \dots, \delta_m). \quad (\text{C.21})$$

This follows our previous definition of prior for individual  $m$  expert parameters. Now, we write the posterior distribution of mixtures of experts model:

$$\begin{aligned}
p(\boldsymbol{\theta}|\mathcal{X}, \mathcal{Y}) &\propto \prod_m^M \prod_{i_m}^{N_m} p(\mathbf{y}_{i_m}|\mathbf{x}_{i_m}, \boldsymbol{\theta}_m) \prod_{m=1}^M \mathcal{N}(\boldsymbol{\theta}_m|\mathbf{0}, \mathbf{A}_m), \\
&\propto \prod_m^M \prod_{i_m}^{N_m} p(\mathbf{y}_{i_m}|\mathbf{x}_{i_m}, \boldsymbol{\theta}_m) \mathcal{N}(\boldsymbol{\theta}_m|\mathbf{0}, \mathbf{A}_m), \\
&\propto \prod_m^M p(\boldsymbol{\theta}_m|\mathcal{D}).
\end{aligned} \tag{C.22}$$

Similarly, we can also express equation C.22 under the data transformation as:

$$p(\boldsymbol{\theta}|\mathcal{X}, \tilde{\mathcal{Y}}) \propto \prod_m^M p(\boldsymbol{\theta}_m|\tilde{\mathcal{D}}). \tag{C.23}$$

This holds as the gating network and the input data  $\mathcal{X}$  are kept the same, and the data transformation is defined on transformed output by the Jacobian of neural networks. Now, as we have already shown with the first Lemma that individual experts have the equivalent posterior distribution, it also follows that  $p(\boldsymbol{\theta}|\mathcal{X}, \mathcal{Y})$  and  $p(\boldsymbol{\theta}|\mathcal{X}, \tilde{\mathcal{Y}})$  are equivalent. We note that the hard gating network enables this step, making it into a valid probability distribution under reverse engineering the densities.

This completes the proof.  $\square$

#### C.1.4. Proof of Lemma C.1.7

**Proposition C.1.5. (Equivalence in Input-Output Relationships)** Let  $f_{\boldsymbol{\theta}}$  be a deterministic deep neural network with the input-prediction set given by  $\{(\mathbf{x}_i, f_{\boldsymbol{\theta}}(\mathbf{x}_i))\}_{i=1}^N$  for all  $N$  data points. Define a mixture of experts model, with a hard gating function  $g_m(\mathbf{x}_i)$  and the neural network experts  $f_{\boldsymbol{\theta}_m} = f_{\boldsymbol{\theta}}$  for all  $m = 1, \dots, M$  experts. Then, the mixture of experts model has an equivalent input-prediction set to the deterministic deep neural network, given by  $\{(\mathbf{x}_i, f_{\boldsymbol{\theta}}(\mathbf{x}_i))\}_{i=1}^N$ .

*Proof.* The proof follows from the definition of the given mixtures of neural network experts.

From the definition, we have a hard gating function that assigns only a single neural network expert per local subset. This means we do not perform any weighted averaging over the predictions. For the inputs  $\mathbf{x}_i$ , the predictions of the mixtures of neural network experts are given by:

$$\mathbf{y}_i = \sum_{m=1}^M g_m(\mathbf{x}_i) f_{\boldsymbol{\theta}_m}(\mathbf{x}_i), \tag{C.24}$$

for all  $i$  and  $m$ . From our definition, we have  $f_{\boldsymbol{\theta}_m} = f_{\boldsymbol{\theta}}$ . Also,  $g_m(\mathbf{x}_i) = 1$  if  $\mathbf{x}_i$  belongs to the  $m^{\text{th}}$  expert. Otherwise, we have  $g_m(\mathbf{x}_i) = 0$ . This results in  $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N = \{(\mathbf{x}_i, f_{\boldsymbol{\theta}}(\mathbf{x}_i))\}_{i=1}^N$ .

This completes the proof.  $\square$

**Lemma C.1.7. (On Approximation Error)** Let  $f_{\text{GP}}(\mathbf{x})$  be a true stationary process such that  $\mathbf{K}_m(\cdot, \cdot) = \mathbf{K}(\cdot, \cdot) \forall m$ . Then, our approximation to  $\mathbf{K}_{\text{true}}(\mathcal{X}, \mathcal{X})$  is  $\mathbf{K}(\mathbf{x}_i, \mathbf{x}_j) = \mathbb{1}_{\mathbf{c}_i = \mathbf{c}_j} \mathbf{K}(\mathbf{x}_i, \mathbf{x}_j)$  where  $\mathbf{c}_i$  is the expert assignment for  $\mathbf{x}_i$ , indicator function  $\mathbb{1}_{a=b} = 1$  when  $a = b$  and 0 otherwise. Then, the approximation error of the local GP kernel is  $\|\mathbf{K}(\mathcal{X}, \mathcal{X}) - \mathbf{K}_{\text{true}}(\mathcal{X}, \mathcal{X})\|_F^2 = \sum_{ij} \mathbf{K}(\mathbf{x}_i, \mathbf{x}_j)^2 - \sum_{m=1}^M \sum_{ij \in \mathcal{W}_m} \mathbf{K}(\mathbf{x}_i, \mathbf{x}_j)^2$ .

*Proof.* Assume we have ordered sets of points as before, and assume the same kernel matrices  $\mathbf{K}_m(\cdot, \cdot) = \mathbf{K}(\cdot, \cdot) \forall m$ . Now, recall the definition of our gating network that  $g_m(\mathbf{x}) = 1$  in just one coordinate for each input,  $\forall m$ . The generative model, considering the entire data to experts, is then:

$$\begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \\ \vdots \\ \mathbf{y}_M \end{bmatrix} = \text{GP} \left( 0, \begin{bmatrix} \mathbf{K}_1 & 0 & 0 & 0 \\ 0 & \mathbf{K}_2 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & \mathbf{K}_M \end{bmatrix} \right)$$

This explains the resulting approximation, that  $\mathbf{K}(\mathbf{x}_i, \mathbf{x}_j) = \mathbb{1}_{\mathbf{c}_i = \mathbf{c}_j} \mathbf{K}(\mathbf{x}_i, \mathbf{x}_j)$  where  $\mathbf{c}_i$  is the partition for  $\mathbf{x}_i$ ,  $\mathbb{1}_{a=b} = 1$  when  $a = b$  and 0 otherwise. In other words, assuming a MoE-DNN with a single, equivalent DNN results in a block-diagonal approximation in the kernel matrix. Decomposing the Frobenius norm then results in  $\|\mathbf{K}(\mathcal{X}, \mathcal{X}) - \mathbf{K}_{true}(\mathcal{X}, \mathcal{X})\|_F^2 = \sum_{ij} \mathbf{K}(\mathbf{x}_i, \mathbf{x}_j)^2 - \sum_{m=1}^M \sum_{ij \in \mathcal{G}_m} \mathbf{K}(\mathbf{x}_i, \mathbf{x}_j)^2$ .

This completes the proof.  $\square$

## C.2. Discussion on Negative Log Likelihood Calculation

When evaluating different uncertainty estimation approaches with negative log likelihood (NLL) as a metric, we are confronted with the issue of different ways to compute it from the quantities predicted by different models. Some predict only the samples from the output distribution directly, such as Monte-Carlo Dropout, while the others predict the moments such as mean and variance of the output distribution directly based on variance propagation (Postels et al., 2019) or linearization over a parameter point estimate (Foong et al., 2019).

Because the calculation of Gaussian likelihood requires the moments (mean and variance) and the observation variance, we can compute it by either following (Gal & Ghahramani, 2016) to evaluate the NLL of each sample predicted by the model (NLL<sub>samples</sub>, refer to equation C.25):

$$\text{NLL}_{\text{samples}}(\mathbf{x}, \mathbf{y}) = -\log \sum_i^T \exp \left( -\frac{(\hat{f}_i(\mathbf{x}) - \mathbf{y})^2}{2\tau} \right) + \log T + \frac{1}{2} \log 2\pi - \frac{1}{2} \log \tau, \quad (\text{C.25})$$

where  $\tau$  is the observation variance,  $T$  is the number of samples,  $\hat{f}_i$  are the sampled predictions. For using the predicted moments directly (NLL<sub>moments</sub>, refer to equation C.26):

$$\text{NLL}_{\text{moments}}(\mathbf{x}, \mathbf{y}) = -\log \exp \left( -\frac{(f_{\text{mean}}(\mathbf{x}) - \mathbf{y})^2}{2(f_{\text{var}}(\mathbf{x}) + \tau)} \right) + \frac{1}{2} \log 2\pi - \frac{1}{2} \log (f_{\text{var}}(\mathbf{x}) + \tau), \quad (\text{C.26})$$

The principled selection of NLL should depend on the quantities they produce; however, because of the different formulations of them (when evaluating on one data pair  $(\mathbf{x}, \mathbf{y})$ ).

Also,  $f_{\text{mean}}$  and  $f_{\text{var}}$  are the moments of the Gaussian distribution, predicted by certain methods. We note that equation C.26 is an exact formula, while the equation C.25 involves approximations. Therefore, the final calculated values differ in the scale of magnitude as well. In order to facilitate a fair comparison, we select the ones that perform best among them. Nevertheless, the issue on this should raise an alarm or attract more attention when researchers decide to choose NLL as their evaluation metric. This also holds when

practitioners decide to select one uncertainty estimation method for their specific purpose. Unfortunately, other measures of regression uncertainty, such as calibration measures, are subject to the results being different depending on the chosen binning (Wenger et al., 2020).

### C.3. Implementation Details and Additional Results

Implementation details and additional results are presented in this section. We used a GPU cluster that consists of NVIDIA GTX 1080 Ti (Pascal), NVIDIA Titan V (Volta), NVIDIA Titan RTX (Turing) and NVIDIA V100 for our off-line processing of the experiments. The CPU pairs used are respectively, 12-core Intel Xeon W2133, 16-core Intel Xeon, and 32-core AMD CPUs. On a robot, we use NVIDIA Jetson TX2 for evaluation of the run-time, which uses the Dual-Core NVIDIA Denver 2 64-Bit and the Quad-Core ARM Cortex-A57 MPCore CPUs. The board features a 256-core NVIDIA Pascal GPU architecture with 8GB of memory. Communication has been established to other computing components of our robot using ROS.

#### C.3.1. Implementation Details for Laplace Approximation Variants

We compare our method against two Laplace Approximation (MacKay, 1992c) based baselines. The first, referred to simply as *Laplace*, is similar to (Ritter et al., 2018b), except that we focus on the diagonal Fisher information matrix (IM) approximation and only consider all linear layers, including multi-head-attention layers, but omitting convolutional layers. This decision allows for a fairer comparison against the second approach referred to as *rLaplace*, which is a linearized version of the first approach similar to (Foong et al., 2019) where no sampling is required. Linearized Laplace approximation relies on the Generalized Gauss-Newton matrix (GNN) of which we also only consider a diagonal approximation for computational reasons.

Both methods are implemented in Python using the PyTorch framework (Hunt et al., 2020; Shinde et al., 2020). We make use of the same pre-trained networks for all approaches. To compute the diagonal IM and GNN approximations, we loop once over the respective training datasets using the same parameterization as during the training of the network weights. Afterwards, we perform a coarse hyperparameter search for each approach and network architecture to find appropriate multiplicative factors used for scaling the IMs and GNNs prior to inversion. For the standard Laplace approximation approach, we sample 30 weight configurations and perform one forward pass for each sample to obtain the distribution over outputs. For the linear Laplace approximation, we compute one GNN per layer and output dimension to increase the accuracy of the approximation.

#### C.3.2. Implementation Details for MC-Dropout Variants

We take MC dropout Gal & Ghahramani (2016) and its real-time variants Postels et al. (2019) which we name approximate variance propagation as two of the baseline approaches. Because both approaches rely on the pre-trained network with a dropout layer, for inverse dynamic experiments, we insert the dropout into the last layer of the three-layer MLP architecture which can obtain satisfactory results in this way from our preliminary experiments. By referring to the Tensorflow implementation of approximate variance propagation, we re-implement this method with Pytorch. We have verified the implementation by

obtaining similar performance on the same benchmarks (UCI) compared to the original implementation.

Nevertheless, we also find that the current implementation does not serve well for some advanced layers such as multi-head attention and LSTM. Therefore, we need to build up the variance propagation layer for them from scratch. On account of the high complexity of the analytical derivations of the Jacobian matrix for these layers, which is hard and error-prone to implement, we decide to make use of the automatic differentiation functionality of Pytorch to achieve it. This comes at a cost of the running time because of the additional backward operation. Specifically, limited by the only backward-mode automatic differentiation support in Pytorch, we need to loop over the output dimension to obtain each row of the Jacobian matrix, which further reduces the efficiency of this method. However, we believe that by this way we can make use of the full power of it and aim to conduct a more fair and comprehensive comparison with our approach and other baseline methods. To note that, for variance propagation, we propagate the covariance matrix in inverse dynamic experiments and a diagonal covariance matrix. For MC-dropout, 30 samples are used for evaluation.

### C.3.3. Implementation Details for Deep Ensembles

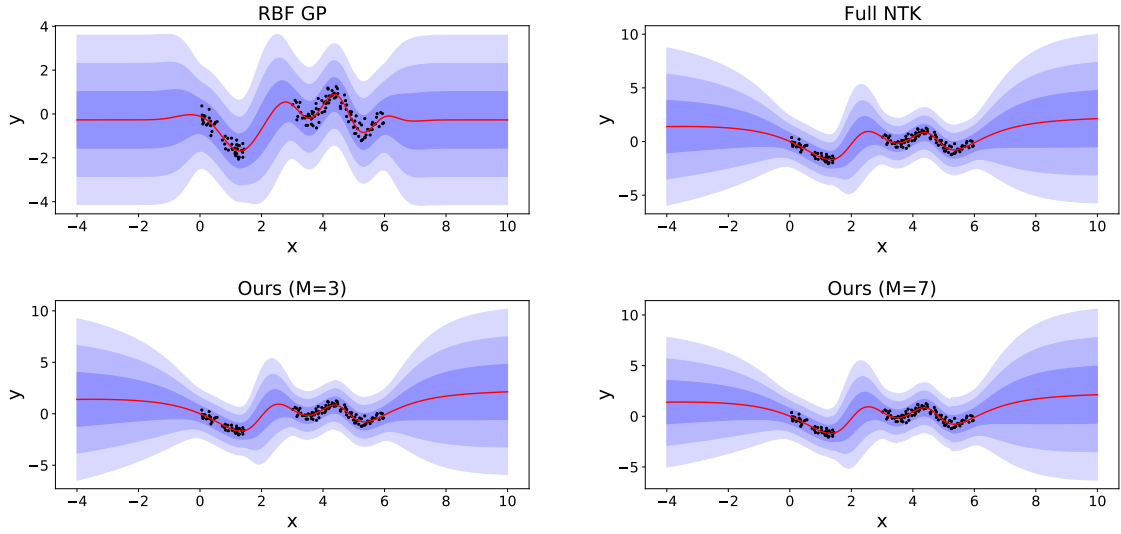
We closely follow the recent protocol of [Sharma et al. \(2021\)](#) for the implementation of deep ensembles ([Lakshminarayanan et al., 2017](#)). In total, five models of identical architecture from random initialization are trained for each dataset, namely SARCOS, KUKA1, and KUKA2, which are repeated over three random seeds. The first member of the ensemble is chosen from the deterministic neural networks, which are used across other baselines.

### C.3.4. Implementation Details and Additional Results for Local GPs

We provide the implementation details of the presented experiments and our method below.

**Toy regression.** Our toy regression experiment used the Snelson dataset, which is often used in GP literature ([Rasmussen, 2003](#)). Due to the small scale, many experimental variables can be carefully controlled for better insights. In this experiment, we trained a single-layer Multi-layer Perceptron (MLP) with 200 units and a tangent activation. The reported figure in the main text used seven GP experts on this dataset. We used the Negative Log Likelihood (NLL) as our loss function and used 10,000 epochs with the Adam optimizer. It used a decay of zero and a learning rate of 0.1. We did not use any memory reduction hyperparameters as the used data and the architectures are small. The only difference between “without patch” and “with patch” settings was the use of the proposed patchwork prior.

**Additional results on toy regression.** We perform additional experiments with GPs as a baseline. The used setup is as follows. We use the same Snelson dataset and a MLP for the toy regression experiments, which is described in the paragraph above. In addition to the comparisons on “in-between” uncertainty estimates ([Foong et al., 2019](#)), we also evaluate on the original Snelson dataset by keeping all the training data. This is to create an additional setup to evaluate if the conclusions still hold. What motivates the Snelson dataset over other experimental setups of our work is the relatively small size of the dataset where full GPs can scale without any further approximations. We note that in GP literature, it is often referred to as large scale problems if the dataset contains more than 10,000 data points where the computational complexity of a full GP is prohibitive. Thus, we compare our approach to the full NTK - the kernel we attempt to approximate in order to improve its applicability beyond 10,000 data points. We also include a GP with

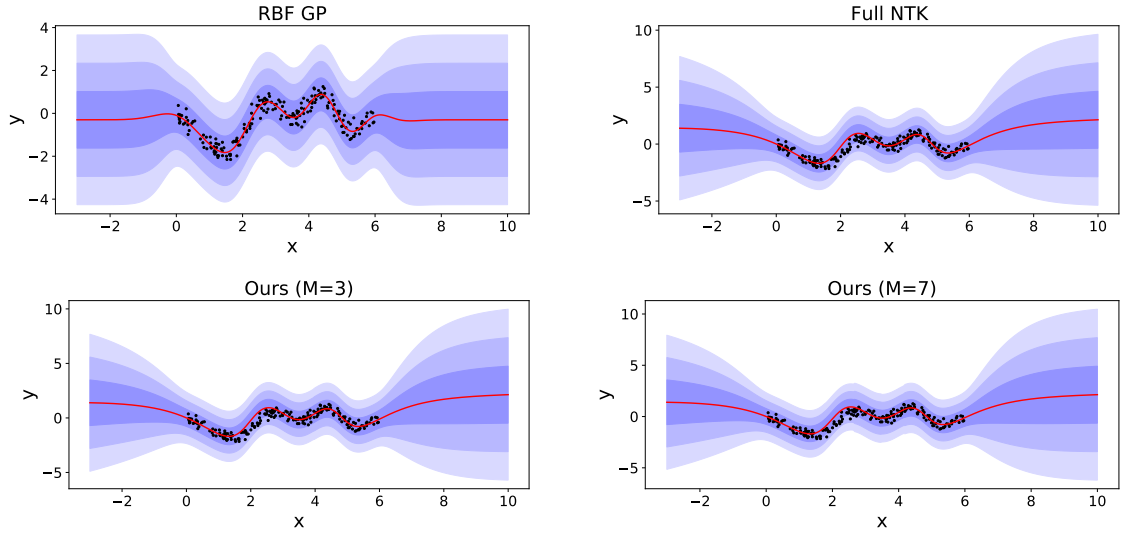


**Figure C.1.:** We visualize predictive uncertainty on Snelson dataset where we remove the training data of certain regime (between  $x = -1$  and  $x = 3$ ) to test the domain shift scenario. Evaluations on the entire Snelson data is depicted in figure C.2. The black dots are train data points, and the red line shows the mean predictions. Blue shades show up to three standard deviations. (Left-Top) We plot a GP with RBF kernel with the original predictions of the GP. (Right-Top) For a validation of the proposed concept, a full NTK without the mixtures of experts approximation is shown. (Left-Bottom) We plot the proposed concept with three GP experts. (Right-Bottom) The increase of experts to seven is visualized. We note that the GP with RBF kernel utilizes a different kernel function and hence, deviations to the GP with the full NTK can be observed both in the predictions and their uncertainty estimates.

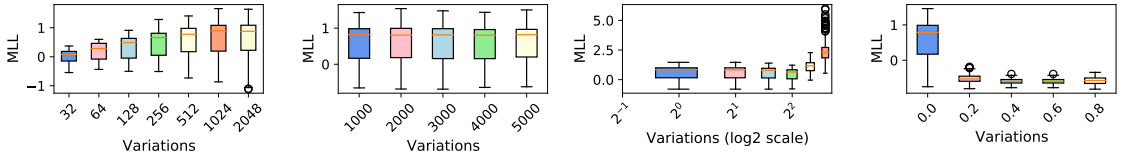
RBF as an additional reference. Lastly, we choose two different settings of local GPs with three experts and seven experts, which is to intuitively support the ablation study on the hyperparameter choices in the work. Figures C.1 and C.2 depict the results.

The following observations can be found in the results. First, the results show that all the considered models qualitatively provide well-calibrated uncertainty estimates, i.e. whenever there is no training data, the models show high uncertainty estimates for both domain-shift and out-of-distribution scenarios. When the regimes where the training data exist, all the models capture the data noise. Second, a GP with RBF kernel provides different predictions to the GPs that utilize the NTK. This behavior is expected as both the models utilize different kernels. Lastly, we find that the proposed concept qualitatively provides similar uncertainty estimates to a GP with the full NTK. This result can be seen as empirical evidence that the proposed approximations to the NTK can provide similar posterior predictive to a GP with the full NTK. The deviations also grow as more approximations are made, i.e. the number of data divisions grows. This observation is similar to the ablation study on the hyperparameter choices, i.e. a larger number of experts can deteriorate the uncertainty estimates, as each GP expert contains a lesser amount of data than the full GP with the full NTK. We refer to our theoretical results, which formalize the proposed approximations to the full NTK.

**Ablation studies.** In our ablation study, we focus on the issue of design choices within our approach. Concretely, there are five hyperparameters, namely, (a) the number of GP experts, (b) the size of the subsets for clustering, (c) the sparsity level of neural and (d) Jacobian pruning (can be considered as a single hyperparameter), and (e) the number of



**Figure C.2.:** We visualize predictive uncertainty on Snelson dataset where we test the uncertainty estimates for out-of-distribution scenarios, and see if the data noise can be captured. Evaluations including the domain shift scenario is depicted in figure C.1. The black dots are train data points, and the red line shows the mean predictions. Blue shades show up to three standard deviations. (Left-Top) We plot a GP with RBF kernel with the original predictions of the GP. (Right-Top) For a validation of the proposed concept, a full NTK without the mixtures of experts approximation is shown. (Left-Bottom) We plot the proposed concept with three GP experts. (Right-Bottom) The increase of experts to seven is visualized. We note that the GP with RBF kernel utilizes a different kernel function and hence, the observed deviations to the GP with the full NTK can be observed both in the predictions and their uncertainty estimates.



**Figure C.3.:** The effects of each hyperparameter is shown with train MLL, by varying them in different steps (Variations), and fixing the others to default settings. Lower the better.

data points for active learning. These design choices influence the uncertainty estimates, as well as the computational complexity. For the empirical study, we examine a regression task using a five-layered, 200 units MLP on the SARCOS dataset [Rasmussen \(2003\)](#). Often used, SARCOS contains 48,933 data points with 21 input values from the joint sensors, and seven joint torque values as targets. We used the SGD optimizer with a learning rate of 0.0001, a momentum of 0.9, and 500 epochs. The loss function was chosen as the mean squared error.

For the ablation studies, we vary each hyperparameter while fixing the remaining ones, and we use the Marginal Log Likelihood (MLL) to evaluate on the training data and the Negative Log Likelihood (NLL) for the testing. The default setup is 512, 5000, 0, and 0 for the number of experts, the subsets for the division step, the level of pruning sparsity, and the proportion of actively selected points respectively. For the number of experts, the variations are: 32, 64, 128, 256, 512, 1024, and 2048. We varied the subset size for the division step from 1000 to 5000 in steps of 5, and the result follows the conclusions of



Chitta et al. (2011). We pruned the Jacobian from up to 0.00001 percent of the layers: 0.0, 0.3, 0.6, 0.9, 0.99, 0.9999, 0.99999. This is to examine the extreme cases where we keep only a few feature maps of the Jacobian for the GP experts. Lastly, we varied the size of the actively selected points from 0.0 to 0.8 in steps of 0.2. Figure C.3 shows the plot for MLL. We did not see any significant difference in trend when comparing both the NLL and the MLL.

The details of the stopping criterion for the active learning and the Jacobian pruning are as follows, which have been the implementation details adapted in all the relevant experiments. In the active learning, we choose the ratio of the most informative data points to select, e.g. 80 percent of the data points. Then, the number of query steps is decided, i.e. we choose three query steps in all the experiments. Selecting the maximization of information as an acquisition function, the active learning process continues until the selected query steps. A similar one-shot pruning strategy is adopted as opposed to an iterative pruning. Similar to active learning, the pre-specified ratio of the pruning applies. Ranking the parameters according to the given metric, the top parameters in the ranking are kept where the numbers are given by the pre-specified ratio. The recipe for the choice of these parameters has been discussed in the ablation study on hyperparameters.

**Learning inverse dynamics of a manipulator.** Now, we consider the task of robot inverse dynamics learning, and evaluate our approach against existing methods for uncertainty estimation. The goal herein is to learn a mapping from the joint positions, velocities, and accelerations to torques  $\tau$  (Nm) for all seven joints. Again, our goal is not to obtain the most accurate prediction, but to estimate predictive uncertainty. A five-layered MLP is trained on SARCOS, KUKA1, KUKA2 with the above mentioned data pairs from SARCOS and KUKA robot arms respectively. We use the test/training split ratio of 1/9 for SARCOS, KUKA1, and KUKA2, and a ratio of 1/99 for the KUKA SIM dataset (used for testing). Besides the nominal settings, to evaluate our approach in a more realistic and challenging way, an OOD scenario, where the models are evaluated on a completely different dataset from training (e.g. train on SARCOS, and test on KUKA1), is conducted. The results are averaged over three random seeds for the error bars of the numerical results.

The used hyperparameters are as follows. For SARCOS, we use ten number of experts, 5000 subsets, sparsity levels of 0.5 and 0.9 for pre-pruning and post-pruning respectively, and use 0.3 for the level of active points. For KUKA, we use 50 Nr. experts, and keep the other hyperparameters the same as SARCOS. We believe that there is still room for better hyperparameter tuning. All GPU memory was not used, as the current implementation can fail due to some GP experts having more data points than others. While SARCOS contains  $\sim 50k$  data pairs of  $100hz$  rhythmic motions, KUKA1 and KUKA2 have  $\sim 200k$  pairs of rhythmic motions at various speeds, respectively. KUKA SIM has a larger size of two million data pairs recorded in simulation. We use the test/training split ratio of 1/9 for SARCOS, KUKA1 and KUKA2, and a ratio of 1/99 for the KUKA SIM dataset.

**Distributed training.** We have tested the scalability of our method. From a theoretic view, local GP scales to infinitely large datasets by creating more and more GP experts. Yet, for the method to be practical, the question is how long the training takes to scale. Scalability is of significant importance in this case, as DNNs operate in the regime of big data, and an exact GP does not scale to such settings. To this end, using the same MLP as before, we train local GP on KUKA SIM dataset (Meier et al., 2014), which contains 1984950 data points. At this scale, an exact GP is clearly not scalable, and many existing solutions resort to incremental learning (Meier et al., 2014). Furthermore, we use a cluster with NVIDIA V100s and perform distributed training. In local GPs, a key benefit over the

other sparse GPs lies in its distributed, local nature, and we can exploit it in practice.

To this end, we use 512 GP experts per joint torque, and use 100 iterations for MLL optimization. For stable training due to the current implementations, we use fewer CPU cores per GPU, while on a single GPU training, all the 32 CPU cores are used. Thus, we see a drop in training time performance when compared to a single GPU. The other hyperparameters used are (a) the subset size of 5000, (b) the active learning parameter of 0.5, (c) and the pruning parameters 0.5 and 0.999 respectively. The training time of an algorithm depends on several factors such as implementation, size of the DNN, and available computational resources. Our attempt is to show how much local GP can scale to large datasets within a reasonable time frame.

**Probabilistic object detection.** We also evaluate our method within the context of an on-going space demo mission for future planetary explorations. The scenario of interests involves a team of heterogeneous robots where the flying system is used for scouting. Once all areas of interest are scouted, the system returns back to the landing units (either on a lander or on a rover). This task of homing is challenging, as the localization algorithm may have accumulated drift or because the roving unit has moved. Therefore, it is necessary to find these particular objects again. Importantly, it is mission critical that the object detector outputs a correct confidence measure. For example, the system should just attempt a landing if it is confident enough to have detected the desired landing systems. This is why we choose to apply our method to the given use-case.

The implementation details are as follows. local GPs with a total of 30 GP experts (three GPs per output) and a single gating network with 5000 subsets are used. We select 80% of the neighboring GPs for the active learning hyperparameter. The same data augmentation as DNNs is used, keeping up to 10000 images of flips and brightness changes. Importantly, for efficiency, we did not use torch.autograd but manually derived the Jacobian of fully connected layers and incorporated them into our forward pass. As the original DETR architecture with ResNet backbones is not designed for real-time applications, we instead use an EfficientNet backbone combined with torch.jit functionality to make the DETR architecture more efficient. We do not consider other popular frameworks of object detection which require several post-processing steps. This is because end-to-end architectures are a preferred way to evaluate uncertainty estimates, without dependencies on how uncertainty estimates are propagated through the post-processing steps. The uncertainties are computed only for the deterministic predictions above a threshold of 0.5. We further note that there are many different ways of designing the probabilistic object detector with our method, and we aim to see the potential of our algorithm in practice.

---

List of Publications

---

**First authorship**

1. **Jongseok Lee**, Ribin Balachandran, Harsimran Singh, Jianxiang Feng, Hrishik Mishra, Marco De Stefano, Rudolph Triebel, Alin Albu Schaeffer and Konstantin Kondak, ‘SPIRIT: Perceptive Shared Autonomy for Aerial Manipulation under Uncertainty in Deep Learning’. Under review at IEEE RA-L.
2. **Jongseok Lee**, Timo Birr, Rudolph Triebel and Tamim Asfour, Stream-based Active Learning for Robust Semantic Perception from Human Instructions. IEEE Robotics and Automation Letter (RA-L), 2025.
3. **Jongseok Lee**, Ribin Balachandran, Konstantin Kondak, Andre Coelho, Marco De Stefano, Matthias Humt, Jianxiang Feng, Tamim Asfour and Rudolph Triebel, Intropective Perception for Long-term Aerial Telem Manipulation with Virtual Reality, IEEE Transactions on Field Robotics (T-FR), 2024.
4. Dominik Schnaus\*, **Jongseok Lee\***, Daniel Cremers and Rudolph Triebel, Learning Expressive Priors for Generalization and Uncertainty Estimation in Neural Networks, In Proc. of the International Conference on Machine Learning (ICML), 2023.
5. **Jongseok Lee\***, Jurrien Olsman\* and Rudolph Triebel, Learning Fluid Flow Visualizations from In-flight Images with Tufts, IEEE Robotics and Automation Letter (RA-L), 2023.
6. **Jongseok Lee**, Ribin Balachandran, Konstantin Kondak, Andre Coelho, Marco De Stefano, Matthias Humt, Jianxiang Feng, Tamim Asfour and Rudolph Triebel, Virtual Reality via Object Pose Estimation and Active Learning: Realizing Telepresence Robots with Aerial Manipulation Capabilities, Field Robotics, 2023.
7. **Jongseok Lee**, Jianxiang Feng, Matthias Humt, Marcus G. Muller and Rudolph Triebel, Trust Your Robots! Predictive Uncertainty Estimation of Neural Networks with Sparse Gaussian Processes, Conference on Robot Learning (CoRL), 2021.

8. **Jongseok Lee**, Matthias Humt, Jianxiang Feng and Rudolph Triebel, Estimating Model Uncertainty of Neural Networks in Sparse Information Form, In Proc. of the International Conference on Machine Learning (ICML), 2020.
9. **Jongseok Lee**, Ribin Balachandran, Yuri S. Sarkisov, Marco De Stefano, Andre Coelho, Kashmira Shinde, Min Jun Kim, Rudolph Triebel, and Konstantin Kondak, Visual-Inertial Telepresence for Aerial Manipulation, In Proc. of the IEEE/RSJ International Conference on Robotics and Automation (ICRA), 2020.
10. **Jongseok Lee**, Tin Muskardin, Cristina Ruiz Paez, Philipp Oettershagen, Thomas Stastny, Inkyu Sa, Roland Siegwart and Konstantin Kondak, Towards Autonomous Stratospheric Flight: A Generic Global System Identification Framework for Fixed-Wing Platforms, In Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2018.

### Co-authorship

1. Martin Schuster et al, Autonomous Planetary Exploration with Robotic Teams - Results of the ARCHES Analog Campaign, Under review at Science Robotics.
2. Ziyuan Qin, **Jongseok Lee**, and Rudolph Triebel, Towards Explaining Uncertainty in Point Cloud Registration, Arxiv, 2024.
3. Jianxiang Feng, **Jongseok Lee**, Simon Geisler, Stephan Guennemann, and Rudolph Triebel, Topology-Matching Normalizing Flows for Out-of-Distribution Detection in Robot Learning, Conference on Robot Learning (CoRL), 2023.
4. Jakob Gawlikowski, Cedrique Rovile Njieutcheu Tassi, Mohsin Ali, **Jongseok Lee**, Matthias Humt, Jianxiang Feng, Anna Kruspe, Rudolph Triebel, Peter Jung, Ribana Roscher, Muhammad Shahzad, Wen Yang, Richard Bamler and Xiao Xiang Zhu, A Survey of Uncertainty in Deep Neural Networks, Artificial Intelligence Review, 2023.
5. Tin Muskardin, Georg Balmer, Linnea Persson, Sven Wlack, Maximilian Laiacker, **Jongseok Lee**, Anibal Ollero, and Konstantin Kondak, Landing of Fixed-Wing Aircraft on Mobile Platforms Accepted to Handbook of Unmanned Aerial Vehicles.
6. Jianxiang Feng, **Jongseok Lee**, Maximilian Durner and Rudolph Triebel, Bayesian Active Learning for Sim-to-Real Robotic Perception, In Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2022.
7. Armin Wedler et al, Finally! Insights into the ARCHES Lunar Planetary Exploration Analogue Campaign on Etna in summer 2022, Proceedings of The International Astronautical Conference (IAC), 2022.
8. Dominik Schnaus, **Jongseok Lee**, Daniel Cremers and Rudolph Triebel, Kronecker-Factored Optimal Curvature, In Bayesian Deep Learning NeurIPS 2021 Workshop, 2021.
9. Armin Wedler et al, First Results from the Multi-Robot, Multi-Partner, Multi-Mission, Planetary Exploration Analogue Campaign on Mt. Etna in Summer 2021, Proceedings of The International Astronautical Conference (IAC), 2021.

- 
10. Andre Coelho, Yuri S. Sarkisov, **Jongseok Lee**, Antonio Franchi, Konstantin Kondak and Christian Ott, Hierarchical Multi-Task Aerial Manipulation with Enhanced Field of View, The 2021 International Conference On Unmanned Aircraft Systems (ICUAS), 2021.
  11. Matthias Humt, **Jongseok Lee** and Rudolph Triebel, Bayesian Optimization Meets Laplace Approximation for Robotic Introspection, In Workshop on Reliable Deployment of Machine Learning for Long-Term Autonomy, the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2020.
  12. Kashmira Shinde, **Jongseok Lee**, Matthias Humt, Aydin Sezgin and Rudolph Triebel, Learning Multiplicative Interactions with Bayesian Neural Networks for Visual-Inertial Odometry, In Workshop on AI for Autonomous Driving (AIAD), the 37th International Conference on Machine Learning (ICML), 2020.
  13. Louis Touko Tcheumadjeu<sup>1a</sup>, Emi Mathews, Arjan Teerhuis, Qinrui Tang<sup>1a</sup>, Jorge Garcia Castano, Marcus Gerhard Muller, Thomas Lobig, Philipp Lutz, **Jongseok Lee**, Robert Kaul, Automated Valet Parking enabled by Internet of Things: A pilot site realization and validation at Brainport, the Netherlands, In 13th ITS European Congress, Brainport, the Netherlands, 2019.
  14. Anastasios Petrou, Daniel Kuster, **Jongseok Lee**, Mirko Meboldt and Marianne Schmid Daners. Comparison of flow estimators for rotary blood pumps: An in-vitro and in-vivo study. *Annals of Biomedical Engineering*, 2018.
  15. Anastasios Petrou, **Jongseok Lee**, Gregor Ochsner, Seraina Dual, Mirko Meboldt, Marianne Schmid Daners, Standardized comparison of selected physiological controllers for rotary blood pumps: In-vitro study, *Artificial Organs*, 42 (3), 2018.



- Aarno, D., Ekvall, S., and Kragic, D. Adaptive virtual fixtures for machine-assisted teleoperation tasks. In *International Conference on Robotics and Automation*, pp. 1139–1144, 2005. 165
- Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., Wicke, M., Yu, Y., and Zheng, X. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pp. 265–283, 2016. 187, 188, 230
- Abril-Pla, O., Andreani, V., Carroll, C., Dong, L., Fannesbeck, C. J., Kochurov, M., Kumar, R., Lao, J., Luhmann, C. C., Martin, O. A., et al. Pymc: a modern, and comprehensive probabilistic programming framework in python. *PeerJ Computer Science*, 9:e1516, 2023. 188
- Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., and Süssstrunk, S. Slc superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2274–2282, 2012. 123
- Achermann, F., Lawrance, N. R., Ranftl, R., Dosovitskiy, A., Chung, J. J., and Siegwart, R. Learning to predict the wind for safe aerial vehicle planning. In *International Conference on Robotics and Automation*, pp. 2311–2317. IEEE, 2019. 113
- Adlam, B., Lee, J., Xiao, L., Pennington, J., and Snoek, J. Exploring the uncertainty properties of neural networks’ implicit priors in the infinite-width limit. In *International Conference on Learning Representations*, 2021. 92
- Agha, A., Otsu, K., Morrell, B., Fan, D. D., Thakker, R., Santamaria-Navarro, A., Kim, S.-K., Bouman, A., Lei, X., Edlund, J. A., Ginting, M. F., Ebadi, K., Anderson, M. O., Pailevanian, T., Terry, E., Wolf, M. T., Tagliabue, A., Vaquero, T. S., Palieri, M., Tepsuporn, S., Chang, Y., Kalantari, A., Chavez, F., Lopez, B. T., Funabiki, N., Miles, G., Touma, T., Buscicchio, A., Tordesillas, J., Alatur, N., Nash, J., Walsh, W., Jung, S., Lee, H., Kanellakis, C., Mayo, J., Harper, S., Kaufmann, M., Dixit, A., Correa, G., Lee, C.-A., Gao, J. L., Merewether, G. B., Maldonado-Contreras, J., Salhotra, G., da Silva, M. S., Ramtoula, B., Fakoorian, S. A., Hatteland, A., Kim, T., Bartlett, T., Stephens, A., Kim, L., Bergh, C. F., Heiden, E., Lew, T., Cauligi, A., Heywood, T., Kramer, A., Leopold, H. A., Choi, C. S., Daftry, S., Toupet, O., Wee, I., Thakur, A., Feras, M., Beltrame, G., Nikolakopoulos, G., Shim, D. H., Carlone, L., and Burdick,



- J. W. Nebula: Quest for robotic autonomy in challenging environments; team costar at the darpa subterranean challenge. *Field Robotics*, 2:1432–1506, 2022. 163
- Aghdam, H. H., Gonzalez-Garcia, A., Weijer, J. v. d., and López, A. M. Active learning for deep detection neural networks. In *International Conference on Computer Vision*, pp. 3672–3680, 2019. 131
- Akcin, O., Unuvar, O., Ure, O., and Chinchali, S. P. Fleet active learning: A submodular maximization approach. In *Conference on Robot Learning*, volume 229, pp. 1378–1399, 06–09 Nov 2023. 101
- Albu-Schäffer, A. O., Haddadin, S., Ott, C., Stemmer, A., Wimböck, T., and Hirzinger, G. The dlr lightweight robot: design and control concepts for robots in human environments. *Ind. Robot*, 34:376–385, 2007. 133
- Alquier, P., Ridgway, J., and Chopin, N. On the properties of variational approximations of gibbs posteriors. *Journal of Machine Learning Research*, 17(1):8374–8414, 2016. 213
- Ambikasaran, S. and O’Neil, M. Fast symmetric factorization of hierarchical matrices with applications. *CoRR*, abs/1405.0223, 2014. 226
- Amini, A., Schwarting, W., Soleimany, A., and Rus, D. Deep evidential regression. *Advances in Neural Information Processing Systems*, 33, 2020. 171
- Ancha, S., Osteen, P. R., and Roy, N. Deep evidential uncertainty estimation for semantic segmentation under out-of-distribution obstacles. In *International Conference on Robotics and Automation*, 2024. 99
- Andrew, A. M. Multiple view geometry in computer vision. *Kybernetes*, 2001. 138
- Antorán, J., Janz, D., Allingham, J. U., Daxberger, E., Barbano, R. R., Nalisnick, E., and Hernández-Lobato, J. M. Adapting the linearised laplace model evidence for modern deep learning. In *International Conference on Machine Learning*, pp. 796–821. PMLR, 2022. 49
- Ardila, D., Kiraly, A. P., Bharadwaj, S., Choi, B., Reicher, J. J., Peng, L., Tse, D., Etemadi, M., Ye, W., Corrado, G., et al. End-to-end lung cancer screening with three-dimensional deep learning on low-dose chest computed tomography. *Nature medicine*, 25(6):954–961, 2019. 21
- Ardywibowo, R., Huo, Z., Wang, Z., Mortazavi, B. J., Huang, S., and Qian, X. Varigrow: Variational architecture growing for task-agnostic continual learning based on bayesian novelty. In *International Conference on Machine Learning*, pp. 865–877. PMLR, 2022. 63
- Arganda-Carreras, I. et al. Crowdsourcing the creation of image segmentation algorithms for connectomics. *Frontiers in neuroanatomy*, 9:142, 2015. 123, 124
- Artigas, J., Balachandran, R., Riecke, C., Stelzer, M., Weber, B., Ryu, J., and Albu-Schaeffer, A. Kontur-2: Force-feedback teleoperation from the international space station. In *International Conference on Robotics and Automation*, pp. 1166–1173, May 2016. 127, 133

- Asfour, T., Regenstein, K., Azad, P., Schroder, J., Bierbaum, A., Vahrenkamp, N., and Dillmann, R. Armar-iii: An integrated humanoid platform for sensory-motor control. In *IEEE-RAS International Conference on Humanoid Robots*, pp. 169–175, 2006. 102
- Asfour, T., Waechter, M., Kaul, L., Rader, S., Weiner, P., Ottenhaus, S., Grimm, R., Zhou, Y., Grotz, M., and Paus, F. Armar-6: A high-performance humanoid for human-robot collaboration in real-world scenarios. *IEEE Robotics and Automation Magazine*, 26(4), 2019. 100, 110, 111
- Azagra, P., Civera, J., and Murillo, A. C. Incremental learning of object models from natural human–robot interactions. *T-ASE*, 17(4):1883–1900, 2020. 101, 102
- Ba, J., Grosse, R., and Martens, J. Distributed second-order optimization using kronecker-factored approximations. In *International Conference on Learning Representations*, 2016. 57, 71, 78
- Babin, P., Giguere, P., and Pomerleau, F. Analysis of robust functions for registration algorithms. In *International Conference on Robotics and Automation*, pp. 1451–1457, 2019. 150, 151
- Bailey, T. and Durrant-Whyte, H. Simultaneous localization and mapping (slam): Part ii. *IEEE Robotics and Automation Magazine*, 13(3):108–117, 2006. 72
- Balachandran, R., Mishra, H., Cappelli, M., Weber, B., Secchi, C., Ott, C., and Albu-Schaeffer, A. Adaptive authority allocation in shared control of robots using bayesian filters. In *International Conference on Robotics and Automation*, pp. 11298–11304, 2020. 161, 165, 167
- Balachandran, R., Mishra, H., Panzirsch, M., and Ott, C. A finite-gain stable multi-agent robot control framework with adaptive authority allocation. In *International Conference on Robotics and Automation*, pp. 1579–1585, 2021a. 133
- Balachandran, R., Panzirsch, M., De Stefano, M., Singh, H., Ott, C., and Albu-Schaeffer, A. Stabilization of user-defined feedback controllers in teleoperation with passive coupling reference. *IEEE Robotics and Automation Letters*, 6(2):3513–3520, 2021b. 127
- Barber, D. and Bishop, C. M. Ensemble Learning for Multi-Layer Networks. In *Advances in Neural Information Processing Systems*, pp. 395–401. MIT Press, 1998. 47
- Barfoot, T. D. *State estimation for robotics*. Cambridge University Press, 2024. 169
- Baydin, A. G., Shao, L., Bhimji, W., Heinrich, L., Naderiparizi, S., Munk, A., Liu, J., Gram-Hansen, B., Louppe, G., Meadows, L., et al. Efficient probabilistic inference in the quest for physics beyond the standard model. *Advances in Neural Information Processing Systems*, 32, 2019. 188
- Bayes, T. An essay towards solving a problem in the doctrine of chances. *Philosophical transactions of the Royal Society of London*, 0(53):370–418, 1763. 55
- Becker, S. and Lecun, Y. Improving the convergence of back-propagation learning with second-order methods. In *Proceedings of the 1988 Connectionist Models Summer School, San Mateo*, pp. 29–37. Morgan Kaufmann, 1989. 78

- Bergamin, F., Moreno-Muñoz, P., Hauberg, S., and Arvanitidis, G. Riemannian laplace approximations for bayesian neural networks. *Advances in Neural Information Processing Systems*, 36:31066–31095, 2023. 48
- Bergna, R., Depeweg, S., Ordonez, S. C., Plenk, J., Cartea, A., and Hernandez-Lobato, J. M. Post-hoc uncertainty quantification in pre-trained neural networks via activation-level gaussian processes. *arXiv preprint arXiv:2502.20966*, 2025. 49, 50, 51
- Bergstra, J. and Bengio, Y. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb):281–305, 2012. 233
- Berkenkamp, F., Turchetta, M., Schoellig, A., and Krause, A. Safe model-based reinforcement learning with stability guarantees. *Advances in Neural Information Processing Systems*, 30, 2017. 24
- Bernard, M. and Kondak, K. Generic slung load transportation system using small size helicopters. In *International Conference on Robotics and Automation*, pp. 3258–3264, May 2009. 127
- Besl, P. J. and McKay, N. D. Method for registration of 3-d shapes. In *Sensor fusion IV: control paradigms and data structures*, volume 1611, pp. 586–606, 1992. 128, 131, 141, 150
- Bindel, D. Power iteration, 2016. URL <https://www.cs.cornell.edu/~protect/unhbox\voldb@x\protect\penalty@M\bindel/class/cs6210-f16/lec/2016-10-17.pdf>. 212
- Bingham, E., Chen, J. P., Jankowiak, M., Obermeyer, F., Pradhan, N., Karaletsos, T., Singh, R., Szerlip, P., Horsfall, P., and Goodman, N. D. Pyro: Deep universal probabilistic programming. *Journal of Machine Learning Research*, 20(1):973–978, 2019. 188
- Bishop, C. M. and Nasrabadi, N. M. *Pattern recognition and machine learning*, volume 4. Springer, 2006. 40, 44, 45, 47, 87, 145, 187, 213, 242
- Blum, A., Hopcroft, J., and Kannan, R. Best-fit subspaces and singular value decomposition (svd). In *Foundations of Data Science*, pp. 29–61. Cambridge University Press, 2020. 212
- Blum, H., Gawel, A., Siegwart, R., and Cadena, C. Modular sensor fusion for semantic segmentation. In *International Conference on Intelligent Robots and Systems*, pp. 3670–3677. IEEE, 2018. 242
- Blum, H., Milano, F., Zurbrügg, R., Siegwart, R., Cadena, C., and Gawel, A. Self-improving semantic perception for indoor localisation. In *Conference on Robot Learning*, 2022. 101
- Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. Weight uncertainty in neural networks. In *International Conference on Machine Learning*, pp. 1613–1622, 2015. 47, 50, 80
- Bodie, K., Brunner, M., Pantic, M., Walser, S., Pfändler, P., Angst, U., Siegwart, R., and Nieto, J. Active interaction force control for contact-based inspection with a fully actuated aerial vehicle. *IEEE Transactions on Robotics*, 37(3):709–722, 2020. 127
- Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., Bernstein, M. S., Bohg, J., Bosselut, A., Brunskill, E., et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021. 67

- Botev, A., Ritter, H., and Barber, D. Practical gauss-newton optimisation for deep learning. In *International Conference on Machine Learning*, volume 70, pp. 557–565. PMLR, 2017. 48, 78, 199
- Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., and Zhang, Q. JAX: composable transformations of Python+NumPy programs, 2018. 187, 188
- Brea, J., Simsek, B., Illing, B., and Gerstner, W. Weight-space symmetry in deep networks gives rise to permutation saddles, connected by equal-loss valleys across the loss landscape. *arXiv preprint arXiv:1907.02911*, 2019. 202
- Brooks, R. A. Symbolic error analysis and robot planning. *International Journal of Robotics Research*, 1(4):29–78, 1982. 23
- Buhmann, J., Burgard, W., Cremers, A. B., Fox, D., Hofmann, T., Schneider, F. E., Strikos, J., and Thrun, S. The mobile robot rhino. *Ai Magazine*, 16(2):31–31, 1995. 24
- Bulatov, Y. Notmnist dataset. *Google (Books/OCR), Tech. Rep.[Online]. Available: <http://yaroslavvb.blogspot.it/2011/09/notmnist-dataset.html>*, 2, 2011. 64
- Burgard, W. Introduction to mobile robotics, 2021. URL <http://ais.informatik.uni-freiburg.de/teaching/ss21/robotics/>. Accessed on August 7, 2025. 33, 34
- Burgard, W., Fox, D., Hennig, D., and Schmidt, T. Estimating the absolute position of a mobile robot using position probability grids. In *Proceedings of the National Conference on Artificial Intelligence*, pp. 896–901, 1996. 23, 24, 106
- Burgard, W., Cremers, A. B., Fox, D., Hähnel, D., Lakemeyer, G., Schulz, D., Steiner, W., and Thrun, S. The museum tour-guide robot rhino. In *Autonome Mobile Systeme 1998: 14. Fachgespräch Karlsruhe, 30. November–1. Dezember 1998*, pp. 245–254. Springer, 1999. 24
- Cacace, J., Orozco-Soto, S. M., Suarez, A., Caballero, A., Orsag, M., Bogdan, S., Vasiljevic, G., Ebeid, E., Rodriguez, J. A. A., and Ollero, A. Safe local aerial manipulation for the installation of devices on power lines: Aerial-core first year results and designs. *Applied Sciences*, 11(13):6220, 2021. 127
- Cacciarelli, D. and Kulahci, M. Active learning for data streams: a survey. *Machine Learning*, 113(1):185–239, 2024. 99, 101
- Cadena, C., Carlone, L., Carrillo, H., Latif, Y., Scaramuzza, D., Neira, J., Reid, I., and Leonard, J. J. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on Robotics*, 32(6):1309–1332, 2016. 23, 182
- Campos, C., Elvira, R., Rodríguez, J. J. G., Montiel, J. M., and Tardós, J. D. Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam. *IEEE Transactions on Robotics*, 37(6):1874–1890, 2021. 185
- Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., and Zagoruyko, S. End-to-end object detection with transformers. In *European Conference on Computer Vision*, pp. 213–229. Springer, 2020a. 121

- Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., and Zagoruyko, S. End-to-end object detection with transformers. In *European Conference on Computer Vision*, pp. 213–229. Springer, 2020b. 95
- Carpenter, B., Gelman, A., Hoffman, M. D., Lee, D., Goodrich, B., Betancourt, M., Brubaker, M., Guo, J., Li, P., and Riddell, A. Stan: A probabilistic programming language. *Journal of statistical software*, 76:1–32, 2017. 187
- Carvalho, C. M., Polson, N. G., and Scott, J. G. Handling sparsity via the horseshoe. In *International Conference on Artificial Intelligence and Statistics*, pp. 73–80. PMLR, 2009. 46, 63
- Carvalho, E. D., Clark, R., Nicastro, A., and Kelly, P. H. Scalable uncertainty for computer vision with functional variational inference. In *Conference on Computer Vision and Pattern Recognition*, pp. 12003–12013, 2020. 46, 92
- Catoni, O. *Pac-Bayesian supervised classification: The thermodynamics of statistical learning*, volume 56 of *Lecture notes, monograph series / Institute of Mathematical Statistics*. Inst. of Math. Statistics, Beachwood, Ohio, 2007. 61, 201, 202, 207, 208
- Charpentier, B., Borchert, O., Zügner, D., Geisler, S., and Günnemann, S. Natural posterior network: Deep bayesian predictive uncertainty for exponential family distributions. In *International Conference on Learning Representations*, 2022. 49
- Cheeseman, P., Smith, R., and Self, M. A stochastic map for uncertain spatial relationships. In *4th international symposium on robotic research*, pp. 467–474. MIT Press Cambridge, 1987. 23
- Chen, L., Suzuki, T., Nonomura, T., and Asai, K. Flow visualization and transient behavior analysis of luminescent mini-tufts after a backward-facing step. *Flow Measurement and Instrumentation*, 71:101657, 2020. 115, 116
- Chen, S.-W., Chou, C.-N., and Chang, E. Y. Bda-pch: Block-diagonal approximation of positive-curvature hessian for training neural networks. *CoRR*, abs/1802.06502, 2018. 77
- Chen, T., Fox, E., and Guestrin, C. Stochastic gradient hamiltonian monte carlo. In *International Conference on Machine Learning*, volume 32-2 of *Proceedings of Machine Learning Research*, pp. 1683–1691. PMLR, 22–24 Jun 2014. 48
- Chitta, R., Jin, R., Havens, T. C., and Jain, A. K. Approximate kernel k-means: Solution to large scale kernel clustering. In *International Conference on Knowledge Discovery and Data Mining*, pp. 895–903, 2011. 90, 257
- Choi, J., Elezi, I., Lee, H.-J., Farabet, C., and Alvarez, J. M. Active learning for deep object detection via probabilistic modeling. In *International Conference on Computer Vision*, pp. 10264–10273, 2021. 131, 147
- Choi, S., Zhou, Q.-Y., and Koltun, V. Robust reconstruction of indoor scenes. In *Conference on Computer Vision and Pattern Recognition*, pp. 5556–5565, 2015. 143
- Choy, C., Dong, W., and Koltun, V. Deep global registration. In *Conference on Computer Vision and Pattern Recognition*, pp. 2514–2523, 2020. 169, 171

- Coelho, A., Sarkisov, Y. S., Lee, J., Balachandran, R., Franchi, A., Kondak, K., and Ott, C. Hierarchical control of redundant aerial manipulators with enhanced field of view. In *International Conference on Unmanned Aircraft Systems*, pp. 994–1002, 2021. 133, 160, 164
- Cohn, D. A., Ghahramani, Z., and Jordan, M. I. Active learning with statistical models. *Journal of Artificial Intelligence Research*, 4:129–145, 1996. 144
- Curtis, A., Matheos, G., Gothoskar, N., Mansinghka, V., Tenenbaum, J. B., Lozano-Perez, T., and Kaelbling, L. P. Partially observable task and motion planning with uncertainty and risk awareness. In *Robotics: Science and Systems*, July 2024. 24
- Da Costa, N., Mucsányi, B., and Hennig, P. Geometric gaussian approximations of probability distributions. *arXiv preprint arXiv:2507.00616*, 2025. 48
- Daftry, S., Zeng, S., Bagnell, J. A., and Hebert, M. Introspective perception: Learning to predict failures in vision systems. In *International Conference on Intelligent Robots and Systems*, pp. 1743–1750. IEEE, 2016. 92
- Dangel, F., Kunstner, F., and Hennig, P. Backpack: Packing more into backprop. In *International Conference on Learning Representations*, 2020. 78
- Daxberger, E., Kristiadi, A., Immer, A., Eschenhagen, R., Bauer, M., and Hennig, P. Laplace redux-effortless bayesian deep learning. *Advances in Neural Information Processing Systems*, 34:20089–20103, 2021a. 45, 48, 50, 55, 57, 58, 59, 60, 188, 205
- Daxberger, E., Nalisnick, E., Allingham, J. U., Antorán, J., and Hernández-Lobato, J. M. Bayesian deep learning via subnetwork inference. In *International Conference on Machine Learning*, pp. 2510–2521. PMLR, 2021b. 48, 49
- De Finetti, B. *Theory of probability: A critical introductory treatment*. John Wiley & Sons, 1970. 36
- De Lange, M., Aljundi, R., Masana, M., Parisot, S., Jia, X., Leonardis, A., Slabaugh, G., and Tuytelaars, T. A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(7):3366–3385, 2022. 63
- Dellaert, F. Factor graphs and gtsam: A hands-on introduction. *Georgia Institute of Technology, Tech. Rep*, 2:4, 2012. 24, 163
- Dellaert, F. and Kaess, M. Square root sam: Simultaneous localization and mapping via square root information smoothing. *International Journal of Robotics Research*, 25(12): 1181–1203, 2006. 23
- Dellaert, F., Fox, D., Burgard, W., and Thrun, S. Monte carlo localization for mobile robots. In *International Conference on Robotics and Automation*, volume 2, pp. 1322–1328. IEEE, 1999. 23
- Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *Conference on Computer Vision and Pattern Recognition*, pp. 248–255. IEEE, 2009. 65, 66

- Deng, Z., Zhou, F., and Zhu, J. Accelerated linearized laplace approximation for bayesian deep learning. *Advances in Neural Information Processing Systems*, 35:2695–2708, 2022. 48, 49
- Denker, J. S. and LeCun, Y. Transforming neural-net output levels to probability distributions. *Advances in Neural Information Processing Systems*, 1991. 48
- Denninger, M. and Triebel, R. Persistent anytime learning of objects from unseen classes. In *International Conference on Intelligent Robots and Systems*, pp. 4075–4082, 2018. 56, 66
- Denninger, M., Winkelbauer, D., Sundermeyer, M., Boerdijk, W., Knauer, M., Strobl, K. H., Humt, M., and Triebel, R. Blenderproc2: A procedural pipeline for photorealistic rendering. *Journal of Open Source Software*, 2023. 169
- Der Kiureghian, A. and Ditlevsen, O. Aleatory or epistemic? does it matter? *Structural safety*, 31(2):105–112, 2009. 35
- Detommaso, G., Gasparin, A., Donini, M., Seeger, M., Wilson, A. G., and Archambeau, C. Fortuna: A library for uncertainty quantification in deep learning. *Journal of Machine Learning Research*, 25(238):1–7, 2024. 188
- Dhillon, I. S., Guan, Y., and Kulis, B. Kernel k-means: spectral clustering and normalized cuts. In *International Conference on Knowledge Discovery and Data Mining*, pp. 551–556, 2004. 90
- Ding, C. and He, X. K-means clustering via principal component analysis. In *International Conference on Machine Learning*, pp. 29, 2004. 90
- Dua, D. and Graff, C. UCI machine learning repository, 2017. 51, 78
- Duane, S., Kennedy, A. D., Pendleton, B. J., and Roweth, D. Hybrid monte carlo. *Physics letters B*, 195(2):216–222, 1987. 187
- Dubey, K. A., J Reddi, S., Williamson, S. A., Póczos, B., Smola, A. J., and Xing, E. P. Variance reduction in stochastic gradient Langevin dynamics. *Advances in Neural Information Processing Systems*, 29, 2016. 47, 48
- Dziugaite, G. K. and Roy, D. M. Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data. *arXiv preprint arXiv:1703.11008*, 2017. 202
- Ebrahimi, S., Elhoseiny, M., Darrell, T., and Rohrbach, M. Uncertainty-guided continual learning with bayesian neural networks. In *International Conference on Learning Representations*, 2020. 63
- Eckart, C. and Young, G. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936. 212
- Elfes, A. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46–57, 1989. 23, 24



- Engelsberger, J., Werner, A., Ott, C., Henze, B., Roa, M. A., Garofalo, G., Burger, R., Beyer, A., Eiberger, O., Schmid, K., et al. Overview of the torque-controlled humanoid robot toro. In *IEEE-RAS International Conference on Humanoid Robots*, pp. 916–923. IEEE, 2014. 184
- Eustice, R. M., Singh, H., and Leonard, J. J. Exactly sparse delayed-state filters for view-based slam. *IEEE Transactions on Robotics*, 22(6):1100–1114, 2006. 72
- Fankhauser, P. and Hutter, M. A universal grid map library: Implementation and use case for rough terrain navigation. In *Robot Operating System (ROS) The Complete Reference (Volume 1)*, pp. 99–120. Springer, 2016. 185
- Farquhar, S., Osborne, M., and Gal, Y. Radial bayesian neural networks: Beyond discrete support in large-scale bayesian deep learning. *International Conference on Artificial Intelligence and Statistics*, 2020a. 46
- Farquhar, S., Smith, L., and Gal, Y. Try depth instead of weight correlations: Mean-field is a less restrictive assumption for deeper networks. *arXiv preprint arXiv:2002.03704*, 2020b. 47
- Felzenszwalb, P. F. and Huttenlocher, D. P. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59:167–181, 2004. 123
- Feng, D., Harakeh, A., Waslander, S. L., and Dietmayer, K. C. J. A review and comparative study on probabilistic object detection in autonomous driving. *IEEE Transactions on Intelligent Transportation Systems*, 23:9961–9980, 2022a. 22, 147
- Feng, J., Durner, M., Márton, Z.-C., Bálint-Benczédi, F., and Triebel, R. Introspective robot perception using smoothed predictions from bayesian neural networks. In *The International Symposium of Robotics Research*, pp. 660–675. Springer, 2019. 78, 92
- Feng, J., Lee, J., Durner, M., and Triebel, R. Bayesian active learning for sim-to-real robotic perception. In *International Conference on Intelligent Robots and Systems*, pp. 10820–10827, 2022b. 32, 101, 107, 115, 118, 147, 152
- Feng, J., Lee, J., Geisler, S., Günnemann, S., and Triebel, R. Topology-matching normalizing flows for out-of-distribution detection in robot learning. In *Conference on Robot Learning*, 2023. 32, 99, 190
- Finn, C., Xu, K., and Levine, S. Probabilistic model-agnostic meta-learning. *Advances in Neural Information Processing Systems*, 31, 2018. 26, 63
- Fischler, M. A. and Bolles, R. C. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6): 381–395, 1981. 139
- Fisher, D. F. and Meyer Jr, R. R. Flow visualization techniques for flight research. In *AGARD Symposium of the Flight Mechanics Panel on Flight Test Techniques*, NAS 1.15: 100455, 1988. 113, 114, 115
- Fontana, M., Zeni, G., and Vantini, S. Conformal prediction: a unified review of theory and new challenges. *Bernoulli*, 29(1):1–23, 2023. 171

- Foong, A. Y., Li, Y., Hernández-Lobato, J. M., and Turner, R. E. 'in-between'uncertainty in bayesian neural networks. *arXiv preprint arXiv:1906.11537*, 2019. 93, 94, 234, 252, 253, 254
- Fort, S. and Scherlis, A. The goldilocks zone: Towards better understanding of neural network loss landscapes. In *AAAI Conference on Artificial Intelligence*, number 01, pp. 3574–3581, 2019. 46
- Fortuin, V. Priors in bayesian deep learning: A review. *International Statistical Review*, 2022. 26, 44, 55, 62, 63
- Fortuin, V., Garriga-Alonso, A., Wenzel, F., Ratsch, G., Turner, R. E., van der Wilk, M., and Aitchison, L. Bayesian neural network priors revisited. In *Symposium on Advances in Approximate Bayesian Inference*, 2021. 62, 63
- Fox, D., Burgard, W., and Thrun, S. Markov localization for mobile robots in dynamic environments. *Journal of Artificial Intelligence Research*, 11:391–427, 1999. 23, 24
- Fox, D., Burgard, W., Kruppa, H., and Thrun, S. A probabilistic approach to collaborative multi-robot localization. *Autonomous Robots*, 8(3):325–344, 2000. 163
- Frazier, P. I. Bayesian optimization. In *Recent advances in optimization and modeling of contemporary problems*, pp. 255–278. Informs, 2018. 23
- Frey, J., Blum, H., Milano, F., Siegwart, R., and Cadena, C. Continual adaptation of semantic segmentation using complementary 2d-3d data representations. *IEEE Robotics and Automation Letters*, 7(4), 2022. 101
- Gal, Y. *Uncertainty in Deep Learning*. PhD thesis, University of Cambridge, 2016. 47, 81, 234
- Gal, Y. and Ghahramani, Z. Bayesian convolutional neural networks with bernoulli approximate variational inference. *arXiv preprint arXiv:1506.02158*, 2015. 215
- Gal, Y. and Ghahramani, Z. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*, pp. 1050–1059, 2016. 29, 47, 50, 66, 67, 85, 86, 88, 92, 94, 95, 99, 108, 118, 122, 131, 152, 165, 246, 252, 253
- Gawlikowski, J., Tassi, C. R. N., Ali, M., Lee, J., Humt, M., Feng, J., Kruspe, A., Triebel, R., Jung, P., Roscher, R., et al. A survey of uncertainty in deep neural networks. *Artificial Intelligence Review*, 56:1513–1589, 2023. 21, 26, 27, 32, 45, 48, 50, 55, 56, 60, 92, 118, 131, 145, 164, 165, 169, 171, 188, 190
- George, T., Laurent, C., Bouthillier, X., Ballas, N., and Vincent, P. Fast approximate natural gradient descent in a kronecker factored eigenbasis. In *Advances in Neural Information Processing Systems*, pp. 9573–9583, 2018. 70, 71, 76, 78, 228
- Germain, P., Bach, F., Lacoste, A., and Lacoste-Julien, S. Pac-bayesian theory meets bayesian inference. *Advances in Neural Information Processing Systems*, 29, 2016. 55, 60, 63, 201, 213

- Ghosh, S., Yao, J., and Doshi-Velez Finale. Structured variational learning of bayesian neural networks with horseshoe priors. *International Conference on Machine Learning*, pp. 1744–1753, 2018. 63
- Gibbs, M. N. *Bayesian Gaussian processes for regression and classification*. PhD thesis, University of Cambridge Doctoral Disertation, 1998. 44
- Giubilato, R., Le Gentil, C., Vayugundla, M., Schuster, M. J., Vidal-Calleja, T., and Triebel, R. Gpgm-slam: a robust slam system for unstructured planetary environments with gaussian process gradient maps. *Field Robotics*, 2:1721–1753, 2022. 182
- Gneiting, T. and Raftery, A. E. Strictly proper scoring rules, prediction, and estimation. *Journal of the American statistical Association*, 102(477):359–378, 2007. 50
- Golub, G. H. and Reinsch, C. Singular value decomposition and least squares solutions. In *Linear Algebra*, pp. 134–151. Springer, 1971. 78
- Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. *Deep learning*, volume 1. MIT press Cambridge, 2016. 25, 42, 56
- Graves, A. Practical variational inference for neural networks. In *Advances in Neural Information Processing Systems*, pp. 2348–2356. Curran Associates, Inc., 2011. 47, 50, 69
- Grimmett, H., Paul, R., Triebel, R., and Posner, I. Knowing when we don’t know: Introspective classification for mission-critical decision making. In *International Conference on Robotics and Automation*, 2013. 83, 85
- Grimmett, H., Triebel, R., Paul, R., and Posner, I. Introspective classification for robot perception. *International Journal of Robotics Research*, 2015. 85, 92
- Grisetti, G., Kümmerle, R., Stachniss, C., and Burgard, W. A tutorial on graph-based slam. *IEEE Intelligent Transportation Systems Magazine*, 2(4):31–43, 2010. 23
- Gross, S., Ranzato, M., and Szlam, A. Hard mixtures of experts for large scale weakly supervised vision. In *Conference on Computer Vision and Pattern Recognition*, pp. 6865–6873, 2017. 88, 242
- Grosse, R. and Martens, J. A kronecker-factored approximate fisher matrix for convolution layers. *International Conference on Machine Learning*, pp. 573–582, 2016. 48, 200, 204, 215
- Guedj, B. A primer on pac-bayesian learning. In *Proceedings of the French Mathematical Society*, pp. 391–414, Lille, France, 2019. Société Mathématique de France. 60, 61, 201, 202, 207, 208
- Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. Improved training of wasserstein gans. *Advances in Neural Information Processing Systems*, 30, 2017. 49
- Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. On calibration of modern neural networks. In *International Conference on Machine Learning*, pp. 1321–1330, 2017. 51, 69, 78, 81, 85, 89, 190

- Gupta, A. K. and Nagar, D. K. *Matrix variate distributions*, volume 104 of *Monographs and surveys in pure and applied mathematics*. Chapman & Hall/CRC, Boca Raton, Fla., 2000. 55
- Gurău, C., Rao, D., Tong, C. H., and Posner, I. Learn from experience: probabilistic prediction of perception performance to avoid failure. *International Journal of Robotics Research*, 37(9):981–995, 2018. 92
- Gustafsson, F. K., Danelljan, M., and Schon, T. B. Evaluating scalable Bayesian deep learning methods for robust computer vision. In *Conference on Computer Vision and Pattern Recognition Workshops*, pp. 318–319, 2020. 67
- Ha, J.-S., Driess, D., and Toussaint, M. A probabilistic framework for constrained manipulations and task and motion planning under uncertainty. In *International Conference on Robotics and Automation*, pp. 6745–6751. IEEE, 2020. 24
- Haidu, A. and Beetz, M. Automated acquisition of structured, semantic models of manipulation activities from human vr demonstration. In *International Conference on Robotics and Automation*, pp. 9460–9466, 2021. 130
- Hamaza, S., Georgilas, I., Conn, A., Heredia, G., Ollero, A., and Richardson, T. A compact and lightweight aerial manipulator for installation and retrieval of sensors in the environment. *Journal of Field Robotics*, 1(1), 2019. 127
- Hara, T., Sato, T., Ogata, T., and Awano, H. Uncertainty-aware haptic shared control with humanoid robots for flexible object manipulation. *IEEE Robotics and Automation Letters*, 2023. 165
- Harakeh, A., Smart, M., and Waslander, S. L. Bayesod: A bayesian approach for uncertainty estimation in deep object detectors. In *International Conference on Robotics and Automation*, pp. 87–93, 2020. 92, 118, 146, 152
- Harrison, J., Willes, J., and Snoek, J. Variational bayesian last layers. In *International Conference on Learning Representations*, 2024. 49
- He, B., Lakshminarayanan, B., and Teh, Y. W. Bayesian deep ensembles via the neural tangent kernel. *arXiv preprint arXiv:2007.05864*, 2020. 92, 194
- He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Conference on Computer Vision and Pattern Recognition*, pp. 770–778. IEEE, 2016. 45, 65, 66, 189
- He, X., Sun, J., Wang, Y., Huang, D., Bao, H., and Zhou, X. Onepose++: Keypoint-free one-shot object pose estimation without cad models. *Advances in Neural Information Processing Systems*, 35:35103–35115, 2022. 188
- He, Z., Feng, W., Zhao, X., and Lv, Y. 6d pose estimation of objects: Recent technologies and challenges. *Applied Sciences*, 11(1):228, 2021. 131
- Hendrycks, D. and Gimpel, K. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *International Conference on Learning Representations*, 2017. 51

- Hernández-Lobato, J. M. and Adams, R. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *International Conference on Machine Learning*, pp. 1861–1869, 2015. 50, 69, 79, 80
- Higham, N. J. Computing a nearest symmetric positive semidefinite matrix. *Linear Algebra and its Applications*, 103:103–118, 1988. 77
- Hinton, G. E. and van Camp, D. Keeping the neural networks simple by minimizing the description length of the weights. In *Conference on Computational Learning Theory*. ACM Press, 1993. 47, 55, 78
- Hirzinger, G., Brunner, B., Landzettel, K., Sporer, N., Butterfass, J., and Schedl, M. Space robotics - dlr’s telerobotic concepts, lightweight arms and articulated hands. *Autonomous Robots*, 14(2):127–145, 2003. 127
- Hoffman, M. D., Gelman, A., et al. The no-u-turn sampler: adaptively setting path lengths in hamiltonian monte carlo. *Journal of Machine Learning Research*, 15(1):1593–1623, 2014. 187
- Hopfield, J. J. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8):2554–2558, 1982. 21
- Hornung, A., Wurm, K. M., Bennewitz, M., Stachniss, C., and Burgard, W. Octomap: An efficient probabilistic 3d mapping framework based on octrees. *Autonomous Robots*, 34: 189–206, 2013. 23
- Huang, K., Chitrakar, D., Ryden, F., and Chizeck, H. J. Evaluation of haptic guidance virtual fixtures and 3d visualization methods in telemanipulation - a user study. *Intell. Serv. Robotics*, 12:289–301, 2019. 128
- Huang, L., Ruan, S., Xing, Y., and Feng, M. A review of uncertainty quantification in medical image analysis: probabilistic and non-probabilistic methods. *Medical Image Analysis*, pp. 103223, 2024. 22
- Hulin, T., Panzirsch, M., Singh, H., Balachandran, R., Coelho, A., Pereira, A., Weber, B. M., Bechtel, N., Riecke, C., Brunner, B., et al. Model-augmented haptic telemanipulation: Concept, retrospective overview and current use-cases. *Frontiers in Robotics and AI*, 8: 76, 2021. 127
- Hüllermeier, E. and Waegeman, W. Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods. *Machine Learning*, 110(3):457–506, 2021. 88
- Humt, M., Lee, J., and Triebel, R. Bayesian optimization meets laplace approximation for robotic introspection. *IROS Workshop on Long-Term Autonomy*, 2020. 57, 145, 146, 147, 253
- Immer, A., Bauer, M., Fortuin, V., Rätsch, G., and Emtiyaz, K. M. Scalable marginal likelihood estimation for model selection in deep learning. In *International Conference on Machine Learning*, pp. 4563–4573. PMLR, 2021a. 26, 47, 63
- Immer, A., Korzepa, M., and Bauer, M. Improving predictions of Bayesian neural nets via local linearization. In *International Conference on Artificial Intelligence and Statistics*, pp. 703–711. PMLR, 2021b. 48, 49

- Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pp. 448–456, 2015. 132
- Izmailov, P., Maddox, W. J., Kirichenko, P., Garipov, T., Vetrov, D., and Wilson, A. G. Subspace inference for Bayesian deep learning. In *Conference on Uncertainty in Artificial Intelligence*, pp. 1169–1179. PMLR, 2020. 48, 49, 50, 78
- Izmailov, P., Vikram, S., Hoffman, M. D., and Wilson, A. G. G. What are bayesian neural network posteriors really like? In *International Conference on Machine Learning*, pp. 4629–4640. PMLR, 2021. 48
- Jacobs, R. A., Jordan, M. I., Nowlan, S. J., and Hinton, G. E. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991. 85, 88, 242
- Jacot, A., Gabriel, F., and Hongler, C. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in Neural Information Processing Systems*, pp. 8571–8580, 2018. 46, 85, 86, 93, 241
- Jospin, L. V., Buntine, W., Boussaid, F., Laga, H., and Bennamoun, M. Hands-on bayesian neural networks—a tutorial for deep learning users. *ACM Comput. Surv.*, 1(1), 2020. 56
- Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Zidek, A., Potapenko, A., Bridgland, A., Meyer, C., Kohl, S. A. A., Ballard, A. J., Cowie, A., Romera-Paredes, B., Nikolov, S., Jain, R., Adler, J., Back, T., Petersen, S., Reiman, D., Clancy, E., Zielinski, M., Steinegger, M., Pacholska, M., Berghammer, T., Bodenstein, S., Silver, D., Vinyals, O., Senior, A. W., Kavukcuoglu, K., Kohli, P., and Hassabis, D. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021. 21
- Kaelbling, L. P., Littman, M. L., and Cassandra, A. R. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1-2):99–134, 1998. 23, 24
- Kaess, M., Ranganathan, A., and Dellaert, F. isam: Incremental smoothing and mapping. *IEEE Transactions on Robotics*, 24(6):1365–1378, 2008. 24
- Kaess, M., Johannsson, H., Roberts, R., Ila, V., Leonard, J. J., and Dellaert, F. isam2: Incremental smoothing and mapping using the bayes tree. *International Journal of Robotics Research*, 31(2):216–235, 2012. 24, 182
- Kanezaki, A. Unsupervised image segmentation by backpropagation. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 1543–1547. IEEE, 2018. 123
- Kao, T.-C., Jensen, K. T., Bernacchia, A., and Hennequin, G. Natural continual learning: success is a journey, not (just) a destination. *arXiv preprint arXiv:2106.08085*, 2021. 58
- Karrer, M., Kamel, M., Siegwart, R., and Chli, M. Real-time dense surface reconstruction for aerial manipulation. In *International Conference on Intelligent Robots and Systems*, pp. 1601–1608, Oct 2016. 128
- Kazhdan, M., Bolitho, M., and Hoppe, H. Poisson surface reconstruction. In *Eurographics Symposium on Geometry Processing*, volume 7, 2006. 143

- Kendall, A. and Gal, Y. What uncertainties do we need in Bayesian deep learning for computer vision? *Advances in Neural Information Processing Systems*, 30, 2017. 35
- Khan, M. E. E., Immer, A., Abedi, E., and Korzepa, M. Approximate inference turns deep networks into gaussian processes. *Advances in Neural Information Processing Systems*, 32, 2019. 49, 87, 89, 93, 213, 241, 243
- Kheddar, A., Caron, S., Gergondet, P., Comport, A., Tanguy, A., Ott, C., Henze, B., Mesesan, G., Engelsberger, J., Roa, M. A., et al. Humanoid robots in aircraft manufacturing: The airbus use cases. *IEEE Robotics and Automation Magazine*, 26(4):30–45, 2019. 184
- Kim, D. and Oh, P. Y. Toward avatar-drone: A human-embodied drone for aerial manipulation. In *International Conference on Unmanned Aircraft Systems*, pp. 567–574, 2021. 127, 130
- Kim, H., Yoon, J., and Lee, J. Fast ensembling with diffusion schrödinger bridge. In *International Conference on Learning Representations*, 2024. 49
- Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015. 204
- Kingma, D. P., Salimans, T., and Welling, M. Variational dropout and the local reparameterization trick. *Advances in Neural Information Processing Systems*, 28, 2015. 47, 69
- Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A. C., Lo, W.-Y., et al. Segment anything. In *International Conference on Computer Vision*, pp. 4015–4026, 2023. 99, 102
- Kirkpatrick, J., Pascanu, R., Rabinowitz, N. C., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., and Hadsell, R. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences of the United States of America*, 114 13: 3521–3526, 2017. 78
- Kirsch, A., Van Amersfoort, J., and Gal, Y. Batchbald: Efficient and diverse batch acquisition for deep Bayesian active learning. *Advances in Neural Information Processing Systems*, 32, 2019. 107
- Knoblauch, J., Jewson, J., and Damoulas, T. An optimization-centric view on bayes’ rule: Reviewing and generalizing variational inference. *Journal of Machine Learning Research*, 23(132):1–109, 2022. 188, 195
- Koh, P. W., Sagawa, S., Marklund, H., Xie, S. M., Zhang, M., Balsubramani, A., Hu, W., Yasunaga, M., Phillips, R. L., Gao, I., Lee, T., David, E., Stavness, I., Guo, W., Earnshaw, B., Haque, I., Beery, S. M., Leskovec, J., Kundaje, A., Pierson, E., Levine, S., Finn, C., and Liang, P. Wilds: A benchmark of in-the-wild distribution shifts. In *International Conference on Machine Learning*, volume 139, pp. 5637–5664. PMLR, 18–24 Jul 2021. 51



- Kohn, S., Blank, A., Puljiz, D., Zenkel, L., Bieber, O., Hein, B., and Franke, J. Towards a real-time environment reconstruction for vr-based teleoperation through model segmentation. In *International Conference on Intelligent Robots and Systems*, pp. 1–9, 2018. 130
- Kondak, K., Huber, F., Schwarzbach, M., Laiacker, M., Sommer, D., Bejar, M., and Ollero, A. Aerial manipulation robot composed of an autonomous helicopter and a 7 degrees of freedom industrial manipulator. In *International Conference on Robotics and Automation*, pp. 2107–2112, May 2014. 127, 133
- Korattikara Balan, A., Rathod, V., Murphy, K. P., and Welling, M. Bayesian dark knowledge. *Advances in Neural Information Processing Systems*, 28, 2015. 49
- Krause, A. and Hübotter, J. Probabilistic artificial intelligence. *arXiv preprint arXiv:2502.05244*, 2025. 35, 45
- Krishnan, R., Subedar, M., and Tickoo, O. Specifying weight priors in bayesian deep neural networks with empirical bayes. *AAAI Conference on Artificial Intelligence*, 34 (04):4477–4484, 2020. 63
- Kristiadi, A., Hein, M., and Hennig, P. Being Bayesian, even just a bit, fixes overconfidence in relu networks. In *International Conference on Machine Learning*, pp. 5436–5446. PMLR, 2020. 78
- Krizhevsky, A. Learning multiple layers of features from tiny images. Technical report, Canadian Institute for Advanced Research, 2009. 65, 78
- Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 25, 2012. 21, 71, 78
- Kschischang, F. R., Frey, B. J., and Loeliger, H.-A. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2):498–519, 2002. 182
- Kumar, A., Chatterjee, S., and Rai, P. Bayesian structural adaptation for continual learning. In *International Conference on Machine Learning*, pp. 5850–5860. PMLR, 2021. 63
- Kümmerle, R., Triebel, R., Pfaff, P., and Burgard, W. Monte carlo localization in outdoor terrains using multilevel surface maps. *Journal of Field Robotics*, 25(6-7):346–359, 2008. 24
- Kummerle, R., Hahnel, D., Dolgov, D., Thrun, S., and Burgard, W. Autonomous driving in a multi-level parking structure. In *International Conference on Robotics and Automation*, pp. 3395–3400, 2009. 24
- Kümmerle, R., Ruhnke, M., Steder, B., Stachniss, C., and Burgard, W. A navigation system for robots operating in crowded urban environments. In *International Conference on Robotics and Automation*, pp. 3225–3232. IEEE, 2013. 24
- Kunstner, F., Hennig, P., and Balles, L. Limitations of the empirical fisher approximation for natural gradient descent. In *Advances in Neural Information Processing Systems*, pp. 4156–4167, 2019. 231

- Kurth, T., Subramanian, S., Harrington, P., Pathak, J., Mardani, M., Hall, D., Miele, A., Kashinath, K., and Anandkumar, A. Fourcastnet: Accelerating global high-resolution weather forecasting using adaptive fourier neural operators. In *Proceedings of the Platform for Advanced Scientific Computing Conference*. Association for Computing Machinery, 2023. 21
- Kweon, I.-S., Hebert, M., Krotkov, E., and Kanade, T. Terrain mapping for a roving planetary explorer. In *International Conference on Robotics and Automation*, pp. 997–1002. IEEE, 1989. 23
- Lai, K., Bo, L., Ren, X., and Fox, D. A large-scale hierarchical multi-view rgb-d object dataset. In *International Conference on Robotics and Automation*, pp. 1817–1824. IEEE, 2011. 66
- Laiacker, M., Huber, F., and Kondak, K. High accuracy visual servoing for aerial manipulation using a 7 degrees of freedom industrial manipulator. In *International Conference on Intelligent Robots and Systems*, pp. 1631–1636, Oct 2016. 131, 138, 140
- Lakshminarayanan, B., Pritzel, A., and Blundell, C. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in Neural Information Processing Systems*, 30, 2017. 49, 50, 67, 78, 81, 85, 88, 95, 99, 108, 132, 152, 165, 216, 232, 246, 254
- Laurent, O., Lafage, A., Tartaglione, E., Daniel, G., Martinez, J.-M., Bursuc, A., Franchi, G., et al. Packed-ensembles for efficient uncertainty estimation. In *International Conference on Learning Representations*, pp. 1–11, 2023. 49
- Lawrence, N., Seeger, M., and Herbrich, R. Fast sparse gaussian process methods: The informative vector machine. *Advances in Neural Information Processing Systems*, 15, 2002. 91
- Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pp. 2278–2324, 1998. 64, 78
- LeCun, Y., Bengio, Y., and Hinton, G. Deep learning. *Nature*, 521(7553):436–444, 2015. 21
- Lee, J., Bahri, Y., Novak, R., Schoenholz, S., Pennington, J., and Sohl-dickstein, J. Deep neural networks as gaussian processes. In *International Conference on Learning Representations*, 2018a. 93, 241
- Lee, J., Muskardin, T., Pacz, C. R., Oettershagen, P., Stastny, T., Sa, I., Siegwart, R., and Kondak, K. Towards autonomous stratospheric flight: A generic global system identification framework for fixed-wing platforms. In *International Conference on Intelligent Robots and Systems*, pp. 6233–6240. IEEE, 2018b. 92, 116
- Lee, J., Balachandran, R., Sarkisov, Y. S., De Stefano, M., Coelho, A., Shinde, K., Kim, M. J., Triebel, R., and Kondak, K. Visual-inertial telepresence for aerial manipulation. In *International Conference on Robotics and Automation*, pp. 1222–1229, 2020a. 32, 92, 160, 164, 168, 169, 192
- Lee, J., Humt, M., Feng, J., and Triebel, R. Estimating model uncertainty of neural networks in sparse information form. In *International Conference on Machine Learning*, pp. 5702–5713, 2020b. 32, 48, 49, 50, 57, 59, 60, 94, 103, 105, 108, 116, 118, 122, 132, 145, 146, 147, 159, 192

- Lee, J., Feng, J., Humt, M., Müller, M. G., and Triebel, R. Trust your robots! predictive uncertainty estimation of neural networks with sparse gaussian processes. In *Conference on Robot Learning*, 2021. 32, 48, 49, 50, 99, 161, 165, 170, 192, 214
- Lee, J., Olsman, W., and Triebel, R. Learning fluid flow visualizations from in-flight images with tufts. *Ieee Robotics and Automation Letters*, 8(6):3677–3684, 2023. 32, 192
- Lee, J., Balachandran, R., Kondak, K., Coelho, A., Stefano, M. d., Humt, M., Feng, J., Asfour, T., and Triebel, R. Introspective perception for long-term aerial telemanipulation with virtual reality. *T-FR*, 1:360–393, 2024. 32, 164, 168, 192
- Lee, J., Balachandran, R., Singh, H., Mishra, H., De Stefano, M., Triebel, R., Albu-Schaeffer, A., and Kondak, K. Spirit: Shared autonomy for robust aerial manipulation under uncertainty in deep learning. *IEEE Robotics and Automation Letters*, 2025a. 32, 192
- Lee, J., Birr, T., Triebel, R., and Asfour, T. Stream-based active learning for robust semantic perception from human instructions. *IEEE Robotics and Automation Letters*, 2025b. 32, 192
- Lehmann, N., Gottschling, N. M., Gawlikowski, J., Stewart, A. J., Depeweg, S., and Nalisnick, E. Lightning uq box: Uncertainty quantification for neural networks. *Journal of Machine Learning Research*, 26(54):1–7, 2025. 188
- Li, Y. and Gal, Y. Dropout inference in bayesian neural networks with alpha-divergences. In *International Conference on Machine Learning*, 2017. 47, 48
- Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P. Focal loss for dense object detection. In *International Conference on Computer Vision*, pp. 2980–2988, 2017. 128, 146, 152
- Liu, C. and Shen, S. An augmented reality interaction interface for autonomous drone. In *International Conference on Intelligent Robots and Systems*, pp. 11419–11424, 2020. 130
- Liu, C., Jiang, R., Wei, D., Yang, C., Li, Y., Wang, F., and Yuan, X. Deep learning approaches in flow visualization. *Advances in Aerodynamics*, 4(1):17, 2022. 113
- Liu, H., Ong, Y.-S., Shen, X., and Cai, J. When gaussian process meets big data: A review of scalable gps. *IEEE Transactions on Neural Networks and Learning Systems*, 2020a. 88
- Liu, J. Z., Lin, Z., Padhy, S., Tran, D., Bedrax-Weiss, T., and Lakshminarayanan, B. Simple and principled uncertainty estimation with deterministic deep learning via distance awareness. *arXiv preprint arXiv:2006.10108*, 2020b. 49, 50, 195
- Louizos, C. and Welling, M. Multiplicative normalizing flows for variational Bayesian neural networks. In *International Conference on Machine Learning*, pp. 2218–2227. PMLR, 2017. 50, 51
- Louizos, C. and Welling Max. Structured and efficient variational deep learning with matrix gaussian posteriors. *International Conference on Machine Learning*, pp. 1708–1716, 2016. 46, 47, 63, 69
- Louizos, C., Ullrich, K., and Welling, M. Bayesian compression for deep learning. *Advances in Neural Information Processing Systems*, 30, 2017. 26, 46

- Lu, Z., Ie, E., and Sha, F. Uncertainty estimation with infinitesimal jackknife, its distribution and mean-field approximation. *CoRR*, abs/2006.07584, 2020. 89, 247
- Lutz, P., Müller, M. G., Maier, M., Stoneman, S., Tomić, T., von Bargaen, I., Schuster, M. J., Steidle, F., Wedler, A., Stürzl, W., et al. Ardea - an mav with skills for future planetary missions. *Journal of Field Robotics*, 37(4):515–551, 2020. 86, 92, 134
- MacKay, D. Bayesian model comparison and backprop nets. *Advances in Neural Information Processing Systems*, 4, 1992a. 48
- MacKay, D. J. Information-based objective functions for active data selection. *Neural Computation*, 4(4):590–604, 1992b. 44, 48, 49, 81, 87, 92, 94, 131, 147, 231, 234, 243
- MacKay, D. J. A practical bayesian framework for backpropagation networks. *Neural Computation*, 4(3):448–472, 1992c. 44, 48, 55, 57, 70, 78, 80, 242, 253
- Maddox, W. J., Garipov, T., Izmailov, P., Vetrov, D., and Wilson, A. G. Fast uncertainty estimates and bayesian model averaging of dnns. In *Uncertainty in Deep Learning Workshop at UAI*, 2018. 78
- Maddox, W. J., Izmailov, P., Garipov, T., Vetrov, D. P., and Wilson, A. G. A simple baseline for bayesian uncertainty in deep learning. *Advances in Neural Information Processing Systems*, 32, 2019. 48, 50, 63, 78, 82, 83, 233
- Mae, Y., Kumagai, W., and Kanamori, T. Uncertainty propagation for dropout-based bayesian neural networks. *Neural Networks*, 144:394–406, 2021. 48
- Maken, F. A., Ramos, F., and Ott, L. Stein icp for uncertainty estimation in point cloud matching. *IEEE Robotics and Automation Letters*, 7(2):1063–1070, 2021. 189
- Maken, F. A., Ramos, F., and Ott, L. Bayesian iterative closest point for mobile robot localization. *International Journal of Robotics Research*, 41(9-10):851–874, 2022. 189
- Malyuta, D., Brommer, C., Hentzen, D., Stastny, T., Siegwart, R., and Brockers, R. Long-duration fully autonomous operation of rotorcraft unmanned aerial systems for remote-sensing data acquisition. *Journal of Field Robotics*, 37(1):137–157, 2020. 131, 138, 140
- Martens, J. New insights and perspectives on the natural gradient method. *arXiv preprint arXiv:1412.1193*, 2014. 199, 204
- Martens, J. and Grosse, R. Optimizing neural networks with kronecker-factored approximate curvature. In *International Conference on Machine Learning*, pp. 2408–2417. PMLR, 2015. 57, 59, 71, 73, 76, 78, 199, 200, 204, 230, 231, 242
- Masone, C., Mohammadi, M., Robuffo Giordano, P., and Franchi, A. Shared planning and control for mobile robots with integral haptic feedback. *International Journal of Robotics Research*, 37(11):1395–1420, 2018. 127
- McAllester, D. Simplified pac-bayesian margin bounds. In *Learning Theory and Kernel Machines*, pp. 203–215. Springer, Berlin, Heidelberg, 2003a. 201
- McAllester, D. A. Pac-bayesian model averaging. In *Conference on Computational Learning Theory*. ACM Press, 1999a. 201

- McAllester, D. A. Some pac-bayesian theorems. *Machine Learning*, 37(3):355–363, 1999b. 55, 61, 63, 201
- McAllester, D. A. Pac-bayesian stochastic model selection. *Machine Learning*, 51(1):5–21, 2003b. 201
- Mees, O., Eitel, A., and Burgard, W. Choosing smartly: Adaptive multimodal fusion for object detection in changing environments. In *International Conference on Intelligent Robots and Systems*, pp. 151–156. IEEE, 2016. 242
- Meier, F., Hennig, P., and Schaal, S. Incremental local gaussian regression. In *Advances in Neural Information Processing Systems*, pp. 972–980, 2014. 88, 94, 257
- Mesanan, G., Schuller, R., Engelsberger, J., Roa, M. A., Lee, J., Ott, C., and Albu-Schäffer, A. Motion planning for humanoid locomotion: Applications to homelike environments. *IEEE Robotics and Automation Magazine*, 2025. 184, 185
- Mishkin, A., Kunstner, F., Nielsen, D., Schmidt, M. W., and Khan, M. E. SLANG: fast structured covariance approximations for bayesian deep learning with natural gradient. In *Advances in Neural Information Processing Systems*, pp. 6248–6258, 2018. 78
- Mittal, S., Tatarchenko, M., and Brox, T. Semi-supervised semantic segmentation with high-and low-level consistency. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(4):1369–1379, 2019. 116
- Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations*, 2018. 49
- Montemerlo, M., Thrun, S., Koller, D., and Wegbreit, B. Fastslam: a factored solution to the simultaneous localization and mapping problem. In *National Conference on Artificial Intelligence*, pp. 593–598, USA, 2002. American Association for Artificial Intelligence. 23
- Morrell, B., Thakker, R., Santamaria Navarro, À., Bouman, A., Lei, X., Edlund, J., Pailevanian, T., Vaquero, T. S., Chang, Y. L., Touma, T., et al. Nebula: Team costarâ€™s robotic autonomy solution that won phase ii of darpa subterranean challenge. *Field robotics*, 2:1432–1506, 2022. 25
- Mühlbauer, M., Hulin, T., Weber, B., Calinon, S., Stulp, F., Albu-Schäffer, A., and Silvério, J. A probabilistic approach to multi-modal adaptive virtual fixtures. *IEEE Robotics and Automation Letters*, 2024. 165
- Mukhoti, J., Stenettorp, P., and Gal, Y. On the importance of strong baselines in bayesian deep learning. *CoRR*, abs/1811.09385, 2018. 234
- Mund, D., Triebel, R., and Cremers, D. Active online confidence boosting for efficient object classification. In *International Conference on Robotics and Automation*, pp. 1367–1373, 2015. 83, 101, 131
- Murphy, K. P. *Probabilistic machine learning: an introduction*. MIT press, 2022. 37
- Muskardin, T., Balmer, G., Persson, L., Wlach, S., Laiacker, M., Ollero, A., and Kondak, K. A novel landing system to increase payload capacity and operational availability of high altitude long endurance uavs. *Journal of Intelligent Robotic Systems*, pp. 1–22, 2017. 113

- Nado, Z., Band, N., Collier, M., Djolonga, J., Dusenberry, M. W., Farquhar, S., Feng, Q., Filos, A., Havasi, M., Jenatton, R., et al. Uncertainty baselines: Benchmarks for uncertainty & robustness in deep learning. *arXiv preprint arXiv:2106.04015*, 2021. 50, 51, 56, 67, 108
- Nair, V. and Hinton, G. E. Rectified linear units improve restricted boltzmann machines. In *International Conference on Machine Learning*, pp. 807–814. PMLR, 2010. 199
- Nalisnick, E. T. *On priors for Bayesian neural networks*. PhD thesis, University of California, Irvine, 2018. 46
- Narr, A., Triebel, R., and Cremers, D. Stream-based active learning for efficient and adaptive classification of 3d objects. In *International Conference on Robotics and Automation*, pp. 227–233. IEEE, 2016. 101, 131
- Neal, R. M. Priors for infinite networks. In *Bayesian Learning for Neural Networks*, pp. 29–53. Springer, 1996a. 93, 241, 250
- Neal, R. M. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 1996b. 47, 55, 78
- Neal, R. M. et al. MCMC using Hamiltonian dynamics. *Handbook of markov chain monte carlo*, 2(11):2, 2011. 80
- Neklyudov, K., Molchanov, D., Ashukha, A., and Vetrov, D. Variance networks: When expectation does not meet your expectations. In *International Conference on Learning Representations*, 2019. 46
- Neubert, P. and Protzel, P. Compact watershed and preemptive slic: On improving trade-offs of superpixel segmentation algorithms. In *2014 22nd international conference on pattern recognition*, pp. 996–1001. IEEE, 2014. 123
- Nguyen, H. and Alexis, K. Forceful aerial manipulation based on an aerial robotic chain: Hybrid modeling and control. *IEEE Robotics and Automation Letters*, 2021. 165
- Ni, D., Song, A., Xu, X., Li, H., Zhu, C., and Zeng, H. 3d-point-cloud registration and real-world dynamic modelling-based virtual environment building method for teleoperation. *Robotica*, 35(10):1958–1974, 2017. 130
- Nissler, C., Durner, M., Mårton, Z.-C., and Triebel, R. Simultaneous calibration and mapping. In *International Symposium on Experimental Robotics*, Buenos Aires, Argentina, Nov. 2018. 131
- Noseworthy, M., Brand, I., Moses, C., Castro, S., Kaelbling, L. P., Lozano-Pérez, T., and Roy, N. Active learning of abstract plan feasibility. In *Robotics: Science and Systems*, 2021. 101
- Ober, S. W. and Aitchison, L. Global inducing point variational posteriors for bayesian neural networks and deep gaussian processes. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8248–8259, 18–24 Jul 2021. 46
- Oh, C., Adamczewski, K., and Park, M. Radial and directional posteriors for bayesian neural networks. In *AAAI Conference on Artificial Intelligence*, number 04, 2020. 46

- Oh, P., Sohn, K., Jang, G., Jun, Y., and Cho, B.-K. Technical overview of team drc-hubo@unlv's approach to the 2015 darpa robotics challenge finals. *Journal of Field Robotics*, 34 (5):874–896, 2017. 130
- Ollero, A., Heredia, G., Franchi, A., Antonelli, G., Kondak, K., Sanfeliu, A., Viguria, A., Martinez-de Dios, J. R., Pierri, F., Cortes, J., Santamaria-Navarro, A., Trujillo Soto, M. A., Balachandran, R., Andrade-Cetto, J., and Rodriguez, A. The aeroarms project: Aerial robots with advanced manipulation capabilities for inspection and maintenance. *IEEE Robotics and Automation Magazine*, 25(4):12–23, Dec 2018. 129, 166
- Ollero, A., Tognon, M., Suarez, A., Lee, D., and Franchi, A. Past, present, and future of aerial robotic manipulators. *IEEE Transactions on Robotics*, 38(1):626–645, 2022. 127, 165
- Olsman, W. F. J. Experimental investigation of fenestron noise. *Journal of the American Helicopter Society*, 67, 2022. 113, 116
- Oppel, M. A bayesian approach to on-line learning. In *On-Line Learning in Neural Networks*, pp. 363–378. Cambridge University Press, 1999. 55
- Oquab, M., Darcet, T., Moutakanni, T., Vo, H. V., Szafraniec, M., Khalidov, V., Fernandez, P., HAZIZA, D., Massa, F., El-Nouby, A., Assran, M., Ballas, N., Galuba, W., Howes, R., Huang, P.-Y., Li, S.-W., Misra, I., Rabbat, M., Sharma, V., Synnaeve, G., Xu, H., Jegou, H., Mairal, J., Labatut, P., Joulin, A., and Bojanowski, P. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research Journal*, 2024. 99, 102
- Ortega, L. A., Rodriguez Santana, S., and Hernández-Lobato, D. Variational linearized Laplace approximation for Bayesian deep learning. In *International Conference on Machine Learning*, volume 235, pp. 38815–38836. PMLR, 21–27 Jul 2024. 48, 49
- Osawa, K., Swaroop, S., Khan, M. E. E., Jain, A., Eschenhagen, R., Turner, R. E., and Yokota, R. Practical deep learning with bayesian principles. *Advances in Neural Information Processing Systems*, 32, 2019. 47, 50
- Ovadia, Y., Fertig, E., Ren, J., Nado, Z., Sculley, D., Nowozin, S., Dillon, J., Lakshminarayanan, B., and Snoek, J. Can you trust your model's uncertainty? Evaluating predictive uncertainty under dataset shift. *Advances in Neural Information Processing Systems*, 32, 2019. 50, 51
- Pace, F. D., Gorjup, G., Bai, H., Sanna, A., Liarokapis, M., and Billingham, M. Leveraging enhanced virtual reality methods and environments for efficient, intuitive, and immersive teleoperation of robots. In *International Conference on Robotics and Automation*, pp. 12967–12973, 2021. 128, 130
- Palmieri, J., Di Lillo, P., Sanfeliu, A., and Marino, A. Perception-driven shared control architecture for agricultural robots performing harvesting tasks. In *International Conference on Intelligent Robots and Systems*, 2024. 165
- Park, C. and Apley, D. Patchwork kriging for large-scale gaussian process regression. *Journal of Machine Learning Research*, 19(1):269–311, 2018. 88, 93



- Park, C. and Huang, J. Z. Efficient Computation of Gaussian Process Regression for Large Spatial Data Sets by Patching Local Gaussian Processes. *Journal of Machine Learning Research*, 17(174):1–29, 2016. 91, 93
- Park, J., Zhou, Q.-Y., and Koltun, V. Colored point cloud registration revisited. In *International Conference on Computer Vision*, pp. 143–152, 2017. 131, 141, 150, 151
- Park, J., Nam, G., Kim, H., Yoon, J., and Lee, J. Ensemble distribution distillation via flow matching. In *International Conference on Machine Learning*, 2025. 49
- Park, Y., Kim, C., and Kim, G. Variational Laplace autoencoders. In *International Conference on Machine Learning*, volume 97, pp. 5032–5041. PMLR, 09–15 Jun 2019. 69
- Paskin, M. A. Thin junction tree filters for simultaneous localization and mapping. In *Int. Joint Conf. on Artificial Intelligence*. Citeseer, 2003. 72, 76
- Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32:8026–8037, 2019. 187, 188, 230
- Pearce, T., Foong, A. Y., and Brintrup, A. Structured weight priors for convolutional neural networks. *arXiv preprint arXiv:2007.14235*, 2020. 46
- Pérez-Ortiz, M., Rivasplata, O., Shawe-Taylor, J., and Szepesvári, C. Tighter risk certificates for neural networks. *Journal of Machine Learning Research*, 22(1):10326–10365, 2021. 67
- Phan, D., Pradhan, N., and Jankowiak, M. Composable effects for flexible and accelerated probabilistic programming in numpyro. In *NeurIPS Workshop on Program Transformations 2019 (to appear)*, 2019. 231
- Pleiss, G., Gardner, J., Weinberger, K., and Wilson, A. G. Constant-time predictive distributions for gaussian processes. In *International Conference on Machine Learning*, pp. 4114–4123. PMLR, 2018. 88, 89
- Pohl, C., Hitzler, K., Grimm, R., Zea, A., Hanebeck, U. D., and Asfour, T. Affordance-based grasping and manipulation in real world applications. In *International Conference on Intelligent Robots and Systems*, pp. 9569–9576, 2020. 130
- Pomerleau, F., Colas, F., and Siegwart, R. A review of point cloud registration algorithms for mobile robotics. *Foundations and Trends in Robotics*, 4(1):1–104, 2015. 131
- Ponomareva, P., Trinitatova, D., Fedoseev, A., Kalinov, I., and Tsetserukou, D. Grasplook: a vr-based telemanipulation system with r-cnn-driven augmentation of virtual environment. In *2021 20th International Conference on Advanced Robotics*, pp. 166–171, 2021. 130
- Postels, J., Ferroni, F., Coskun, H., Navab, N., and Tombari, F. Sampling-free epistemic uncertainty estimation using approximated variance propagation. In *International Conference on Computer Vision*, pp. 2931–2940, 2019. 48, 92, 94, 252, 253
- Pourzanjani, A. A., Jiang, R. M., and Petzold, L. R. Improving the identifiability of neural networks for bayesian inference. In *NeurIPS Workshop on Bayesian Deep Learning*, volume 4, pp. 29, 2017. 202

- Puljiz, D., Krebs, F., Bosing, F., and Hein, B. What the hololens maps is your workspace: Fast mapping and set-up of robot cells via head mounted displays and augmented reality. In *International Conference on Intelligent Robots and Systems*, pp. 11445–11451, 2020. 130
- Pumarola, A., Vakhitov, A., Agudo, A., Moreno-Noguer, F., and Sanfeliu, A. Relative localization for aerial manipulation with pl-slam. In *Aerial Robotic Manipulation*, pp. 239–248. Springer, 2019. 128
- Raiola, G., Restrepo, S. S., Chevalier, P., Rodriguez-Ayerbe, P., Lamy, X., Tliba, S., and Stulp, F. Co-manipulation with a library of virtual guiding fixtures. *Autonomous Robots*, 42:1037–1051, 2018. 165
- Ramanagopal, M. S., Anderson, C., Vasudevan, R., and Johnson-Roberson, M. Failing to learn: autonomously identifying perception failures for self-driving cars. *IEEE Robotics and Automation Letters*, 3(4):3860–3867, 2018. 92
- Ramos, F., Possas, R., and Fox, D. Bayessim: Adaptive domain randomization via probabilistic inference for robotics simulators. In *Robotics: Science and Systems*, June 2019. 24
- Ranganath, R., Tang, L., Charlin, L., and Blei, D. Deep exponential families. In *Artificial Intelligence and Statistics*, pp. 762–771. PMLR, 2015. 49
- Rasmussen, C. E. Gaussian processes for machine learning. In *Summer School on Machine Learning*, pp. 63–71. Springer, 2003. 46, 87, 88, 94, 170, 254, 256
- Rasmussen, C. E. and Ghahramani, Z. Infinite mixtures of gaussian process experts. In *Advances in Neural Information Processing Systems*, pp. 881–888, 2002. 242, 246
- Ravichandar, H., Polydoros, A. S., Chernova, S., and Billard, A. Recent advances in robot learning from demonstration. *Annual Review of Control, Robotics, and Autonomous Systems*, 3(1):297–330, 2020. 101
- Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. You only look once: Unified, real-time object detection. In *Conference on Computer Vision and Pattern Recognition*, pp. 779–788, 2016. 142, 189
- Reichlin, A., Vasco, M., and Kragic, D. Walking on the fiber: A simple geometric approximation for bayesian neural networks. *Transactions on Machine Learning Research*, 2025. 48
- Ren, A. Z., Dixit, A., Bodrova, A., Singh, S., Tu, S., Brown, N., Xu, P., Takayama, L., Xia, F., Varley, J., et al. Robots that ask for help: Uncertainty alignment for large language model planners. In *Conference on Robot Learning*, pp. 661–682. PMLR, 2023. 99, 101
- Ren, P., Xiao, Y., Chang, X., Huang, P.-Y., Li, Z., Gupta, B. B., Chen, X., and Wang, X. A survey of deep active learning. *ACM computing surveys*, 54(9):1–40, 2021. 23
- Richter, C. and Roy, N. Safe visual navigation via deep learning and novelty detection. *Robotics: Science and Systems*, 2017. 92

- Ritter, H., Botev, A., and Barber, D. Online structured laplace approximations for overcoming catastrophic forgetting. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc, 2018a. 48, 58, 78, 215, 237
- Ritter, H., Botev, A., and Barber, D. A scalable laplace approximation for neural networks. In *International Conference on Learning Representations*, 2018b. 48, 50, 55, 57, 58, 59, 60, 70, 71, 78, 80, 94, 215, 230, 234, 237, 253
- Robbins, H. E. An empirical bayes approach to statistics. In *Breakthroughs in statistics*, pp. 388–394. Springer, 1992. 63
- Rosenberg, L. B. Virtual fixtures: Perceptual tools for telerobotic manipulation. In *Proceedings of IEEE Virtual Reality Annual International Symposium*, pp. 76–82, 1993. 134
- Rothfuss, J., Fortuin, V., Josifoski, M., and Krause, A. Pacoh: Bayes-optimal meta-learning with pac-guarantees. In *International Conference on Machine Learning*, pp. 9116–9126. PMLR, 2021a. 47, 63
- Rothfuss, J., Heyn, D., Krause, A., et al. Meta-learning reliable priors in the function space. *Advances in Neural Information Processing Systems*, 34:280–293, 2021b. 47, 63
- Rothfuss, J., Sukhija, B., Treven, L., Dörfler, F., Coros, S., and Krause, A. Bridging the sim-to-real gap with bayesian inference. In *International Conference on Intelligent Robots and Systems*, pp. 10784–10791. IEEE, 2024. 47
- Roy, H., Miani, M., Ek, C. H., Hennig, P., Pförtner, M., Tatzel, L., and Hauberg, S. r. Reparameterization invariance in approximate bayesian inference. In *Advances in Neural Information Processing Systems*, volume 37, pp. 8132–8164. Curran Associates, Inc., 2024. 48, 49
- Roy, N., Burgard, W., Fox, D., and Thrun, S. Coastal navigation-mobile robot navigation with uncertainty in dynamic environments. In *International Conference on Robotics and Automation*, volume 1, pp. 35–40. IEEE, 1999. 23, 24, 163
- Roy, S., Unmesh, A., and Namboodiri, V. P. Deep active learning for object detection. In *British Machine Vision Conference*, volume 362, pp. 91, 2018. 147
- Rudner, T. G., Chen, Z., Teh, Y. W., and Gal, Y. Tractable function-space variational inference in bayesian neural networks. *Advances in Neural Information Processing Systems*, 35:22686–22698, 2022a. 46, 194
- Rudner, T. G., Smith, F. B., Feng, Q., Teh, Y. W., and Gal, Y. Continual learning via sequential function-space variational inference. In *International Conference on Machine Learning*, pp. 18871–18887. PMLR, 2022b. 63
- Rudner, T. G., Kapoor, S., Qiu, S., and Wilson, A. G. Function-space regularization in neural networks: A probabilistic perspective. In *International Conference on Machine Learning*, pp. 29275–29290. PMLR, 2023. 47, 50, 63
- Rudner, T. G. J., Pan, X., Li, Y. L., Shwartz-Ziv, R., and Wilson, A. G. Fine-tuning with uncertainty-aware priors makes vision and language foundation models more reliable. In *International Conference on Artificial Intelligence and Statistics*, 2025. 47

- Ruiz, C., Acosta, J., and Ollero, A. Aerodynamic reduced-order volterra model of an ornithopter under high-amplitude flapping. *Aerospace Science and Technology*, 121: 107331, 2022. 113
- Rumelhart, D. E., Hinton, G. E., and Williams, R. J. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986. 21
- Rusinkiewicz, S. and Levoy, M. Efficient variants of the icp algorithm. In *Proceedings third international conference on 3-D digital imaging and modeling*, pp. 145–152, 2001. 131, 141, 150
- Rusu, A. A., Rabinowitz, N. C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., and Hadsell, R. Progressive neural networks. *ArXiv*, abs/1606.04671, 2016. 55, 61, 63, 202
- Rusu, A. A., Večerík, M., Rothörl, T., Heess, N., Pascanu, R., and Hadsell, R. Sim-to-real robot learning from pixels with progressive nets. In *Conference on Robot Learning*, pp. 262–270. PMLR, 2017. 62
- Saeidi, H. and Wang, Y. Incorporating trust and self-confidence analysis in the guidance and control of (semi) autonomous mobile robotic systems. *IEEE Robotics and Automation Letters*, 4(2):239–246, 2018. 165
- Sagun, L., Evci, U., Güney, V. U., Dauphin, Y., and Bottou, L. Empirical analysis of the hessian of over-parametrized neural networks. In *International Conference on Learning Representations*, 2018. 70, 73, 235
- Sanchez-Cuevas, P. J., Ramon-Soria, P., Arrue, B., Ollero, A., and Heredia, G. Robotic system for inspection by contact of bridge beams using uavs. *Sensors*, 19(2), 2019. 127
- Saran, A., Yousefi, S., Krishnamurthy, A., Langford, J., and Ash, J. T. Streaming active learning with deep neural networks. In *International Conference on Machine Learning*, pp. 30005–30021, 2023. 101
- Sarkisov, Y. S., Kim, M. J., Bicego, D., Tsetserukou, D., Ott, C., Franchi, A., and Kondak, K. Development of sam: Cable-suspended aerial manipulator. In *International Conference on Robotics and Automation*, pp. 5323–5329, May 2019. 128, 129, 133
- Sarlin, P.-E., Unagar, A., Larsson, M., Germain, H., Toft, C., Larsson, V., Pollefeys, M., Lepetit, V., Hammarstrand, L., Kahl, F., et al. Back to the feature: Learning robust camera localization from pixels to pose. In *Conference on Computer Vision and Pattern Recognition*, pp. 3247–3257, 2021. 188
- Scannell, A., Mereu, R., Chang, P., Tami, E., Pajarinen, J., and Solin, A. Function-space parameterization of neural networks for sequential learning. In *International Conference on Learning Representations*, 5 2024. 48, 49
- Schiebener, D., Ude, A., Morimoto, J., Asfour, T., and Dillmann, R. Segmentation and learning of unknown objects through physical interaction. In *IEEE-RAS International Conference on Humanoid Robots*, pp. 500–506, 2011. 101, 102
- Schmid, K., Lutz, P., Tomić, T., Mair, E., and Hirschmüller, H. Autonomous vision-based micro air vehicle for indoor and outdoor navigation. *Journal of Field Robotics*, 31(4): 537–570, 2014. 134

- Schmidt, S. S. and Gunnemann, S. Stream-based active learning by exploiting temporal properties in perception with temporal predicted loss. *British Machine Vision Conference*, 2023. 101
- Schmitt, J. and Roth, S. Sampling-free variational inference for neural networks with multiplicative activation noise. In *DAGM German Conference on Pattern Recognition*, pp. 33–47. Springer, 2021. 49
- Schnaus, D., Lee, J., and Triebel, R. Kronecker-factored optimal curvature. In *Bayesian Deep Learning NeurIPS 2021 Workshop*, 2021. 200, 204, 214
- Schnaus, D., Lee, J., Cremers, D., and Triebel, R. Learning expressive priors for generalization and uncertainty estimation in neural networks. In *International Conference on Machine Learning*, 2023. 32, 47, 50, 101, 103, 104, 105, 108, 192
- Schuster, M. J., Schmid, K., Brand, C., and Beetz, M. Distributed stereo vision-based 6d localization and mapping for multi-robot teams. *Journal of Field Robotics*, 36(2): 305–332, 2019. 182, 184
- Schuster, M. J., Müller, M. G., Brunner, S. G., Lehner, H., Lehner, P., Sakagami, R., Dömel, A., Meyer, L., Vodermayr, B., Giubilato, R., et al. The arches space-analogue demonstration mission: towards heterogeneous teams of autonomous robots for collaborative scientific sampling in planetary exploration. *IEEE Robotics and Automation Letters*, 5(4):5315–5322, 2020. 95, 181, 184
- Selvaggio, M., Cognetti, M., Nikolaidis, S., Ivaldi, S., and Siciliano, B. Autonomy in physical human-robot interaction: A brief survey. *IEEE Robotics and Automation Letters*, 6(4): 7989–7996, 2021. 164, 165, 166
- Sensoy, M., Kaplan, L., and Kandemir, M. Evidential deep learning to quantify classification uncertainty. In *Advances in Neural Information Processing Systems*, pp. 3179–3189, 2018. 92
- Settles, B. Active learning. *Synthesis lectures on artificial intelligence and machine learning*, 6(1):1–114, 2012. 23, 26
- Shalev-Shwartz, S. and Ben-David, S. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014. 63
- Sharma, A., Azizan, N., and Pavone, M. Sketching curvature for efficient out-of-distribution detection for deep neural networks. *Conference on Uncertainty in Artificial Intelligence*, 2021. 254
- Shekhovtsov, A. and Flach, B. Feed-forward propagation in probabilistic neural networks with categorical and max layers. In *International Conference on Learning Representations*, 2019. 48
- Shi, L. X., Hu, Z., Zhao, T. Z., Sharma, A., Pertsch, K., Luo, J., Levine, S., and Finn, C. Yell at your robot: Improving on-the-fly from language corrections. *Robotics: Science and Systems*, 2024. 101

- Shinde, K., Lee, J., Humt, M., Sezgin, A., and Triebel, R. Learning multiplicative interactions with bayesian neural networks for visual-inertial odometry. In *Workshop on AI for Autonomous Driving (AIAD), the 37th International Conference on Machine Learning (ICML)*, 2020. 253
- Shwartz-Ziv, R., Goldblum, M., Souri, H., Kapoor, S., Zhu, C., LeCun, Y., and Wilson, A. G. Pre-train your loss: Easy bayesian transfer learning with informative priors. *arXiv preprint arXiv:2205.10279*, 2022. 63
- Simoncelli, E. P. Capturing visual image properties with probabilistic models. In *The essential guide to image processing*, pp. 205–223. Elsevier, 2009. 46
- Singh, S. P. and Hofmann, T. Closed form of the hessian spectrum for some neural networks. In *High-dimensional Learning Dynamics 2024: The Emergence of Structure and Reasoning*, 2024. 70, 73
- Singh, S. P., Bachmann, G., and Hofmann, T. Analytic insights into structure and rank of neural network hessian maps. *Advances in Neural Information Processing Systems*, 34: 23914–23927, 2021. 70, 73
- Singh, S. P., Hofmann, T., and Schölkopf, B. The hessian perspective into the nature of convolutional neural networks. In *International Conference on Machine Learning*, pp. 31930–31968. PMLR, 2023. 70, 73
- Sirohi, K., Marvi, S., Büscher, D., and Burgard, W. Uncertainty-aware panoptic segmentation. *IEEE Robotics and Automation Letters*, 8(5):2629–2636, 2023. 99
- Sodhi, P., Dexheimer, E., Mukadam, M., Anderson, S., and Kaess, M. Leo: Learning energy-based models in factor graph optimization. In *Conference on Robot Learning*, pp. 234–244. PMLR, 2022. 24
- Spiegelhalter, D. J. and Lauritzen, S. L. Sequential updating of conditional probabilities on directed graphical structures. *Networks*, 20(5):579–605, 1990. 44
- Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014. 132
- Steger, C. An unbiased detector of curvilinear structures. *IEEE Transactions on Pattern Recognition and Machine Intelligence*, 20(2):113–125, 1998. 120, 121, 123, 124
- Steinfurth, B., Cura, C., Gehring, J., and Weiss, J. Tuft deflection velocimetry: a simple method to extract quantitative flow field information. *Experiments in Fluids*, 61:1–11, 2020. 113, 114, 115, 116
- Sun, S., Chen, C., and Carin, L. Learning structured weight uncertainty in bayesian neural networks. *Artificial Intelligence and Statistics*, pp. 1283–1292, 2017. 50, 69
- Sun, S., Zhang, G., Shi, J., and Grosse, R. Functional variational bayesian neural networks. In *International Conference on Learning Representations*, 2019. 46, 50, 62, 63, 85, 194
- Sünderhauf, N., Brock, O., Scheirer, W., Hadsell, R., Fox, D., Leitner, J., Upcroft, B., Abbeel, P., Burgard, W., Milford, M., et al. The limits and potentials of deep learning for robotics. *International Journal of Robotics Research*, 37(4-5):405–420, 2018. 26

- Sünderhauf, N., Dayoub, F., Hall, D., Skinner, J., Zhang, H., Carneiro, G., and Corke, P. A probabilistic challenge for object detection. *Nature Machine Intelligence*, 1(9):443–443, 2019. 92
- Sutton, R. S. Reinforcement learning: An introduction. *A Bradford Book*, 2018. 23, 26
- Tan, M., Pang, R., and Le, Q. V. Efficientdet: Scalable and efficient object detection. In *Conference on Computer Vision and Pattern Recognition*, pp. 10781–10790, 2020. 121
- Tang, Z., Jiang, F., Gong, M., Li, H., Wu, Y., Yu, F., Wang, Z., and Wang, M. Skfac: Training neural networks with faster kronecker-factored approximate curvature. In *Conference on Computer Vision and Pattern Recognition*, pp. 13479–13487, 2021. 200
- Teh, Y. W., Thiery, A. H., and Vollmer, S. J. Consistency and fluctuations for stochastic gradient Langevin dynamics. *Journal of Machine Learning Research*, 17, 2016. 48
- Thomas, H., Katharina, H., Carsten, P., Chun-Yi, S., Subhash, R., and Honghai, L. Interactive features for robot viewers. In *Intelligent Robotics and Applications*, volume 7508, 2012. 133, 134, 135
- Thrun, S. and Liu, Y. Multi-robot slam with sparse extended information filters. In *Robotics Research. The Eleventh International Symposium*, pp. 254–266. Springer, 2005. 72
- Thrun, S. and Mitchell, T. M. Lifelong robot learning. *Robotics and Autonomous Systems*, 15(1-2):25–46, 1995. 63
- Thrun, S. and Pratt, L. Learning to learn: Introduction and overview. In *Learning to learn*, pp. 3–17. Springer, 1998. 63
- Thrun, S., Beetz, M., Bennewitz, M., Burgard, W., Cremers, A. B., Dellaert, F., Fox, D., Haehnel, D., Rosenberg, C., Roy, N., et al. Probabilistic algorithms and the interactive museum tour-guide robot minerva. *International Journal of Robotics Research*, 19(11): 972–999, 2000. 24, 161, 163
- Thrun, S., Fox, D., and Burgard, W. Probabilistic robotics. *Communications of the ACM*, 45(3):52–57, 2002. 23, 25, 34, 85, 92
- Thrun, S., Liu, Y., Koller, D., Ng, A. Y., Ghahramani, Z., and Durrant-Whyte, H. Simultaneous localization and mapping with sparse extended information filters. *International Journal of Robotics Research*, 23(7-8):693–716, 2004. 23, 69, 72, 78, 231
- Tishby, N., Levin, E., and Solla, S. A. Consistent inference of probabilities in layered networks: Predictions and generalization. In *International Joint Conference on Neural Networks*, volume 2, pp. 403–409. IEEE, 1989. 46
- Tran, D., Hoffman, M. D., Saurous, R. A., Brevdo, E., Murphy, K., and Blei, D. M. Deep probabilistic programming. In *International Conference on Learning Representations*, 2017. 188
- Tran, D., Liu, J., Dusenberry, M. W., Phan, D., Collier, M., Ren, J., Han, K., Wang, Z., Mariet, Z., Hu, H., et al. Plex: Towards reliability using pretrained large model extensions. *arXiv preprint arXiv:2207.07411*, 2022. 50, 51, 56, 63, 66, 67, 216



- Tresp, V. Mixtures of gaussian processes. In *Advances in Neural Information Processing Systems*, pp. 654–660, 2001. 85, 88, 242, 246
- Triebel, R., Pfaff, P., and Burgard, W. Multi-level surface maps for outdoor terrain mapping and loop closing. In *IEEE/RSJ*, pp. 2276–2282. IEEE, 2006. 23, 24
- Triebel, R., Grimmer, H., Paul, R., and Posner, I. Driven learning for driving: How introspection improves semantic mapping. In *Robotics Research*, pp. 449–465. Springer, 2016. 91, 101, 106
- Trinh, S., Spindler, F., Marchand, E., and Chaumette, F. A modular framework for model-based visual tracking using edge, texture and depth features. In *International Conference on Intelligent Robots and Systems*, pp. 89–96. IEEE, 2018. 184
- Trinh, T. H., Wu, Y., Le, Q. V., He, H., and Luong, T. Solving olympiad geometry without human demonstrations. *Nature*, 625(7995):476–482, 2024. 21
- Trujillo, M. A., Martínez-de Dios, J. R., Martín, C., Viguria, A., and Ollero, A. Novel aerial manipulator for accurate and robust industrial ndt contact inspection: A new tool for the oil and gas inspection industry. *Sensors*, 19(6), 2019. 127
- Vakhitov, A., Ferraz, L., Agudo, A., and Moreno-Noguer, F. Uncertainty-aware camera pose estimation from points and lines. In *Conference on Computer Vision and Pattern Recognition*, pp. 4659–4668, 2021. 189
- Valipour, S., Perez, C., and Jagersand, M. Incremental learning for robot perception through hri. In *International Conference on Intelligent Robots and Systems*, pp. 2772–2777, 2017. 101, 102
- Van Amersfoort, J., Smith, L., Teh, Y. W., and Gal, Y. Uncertainty estimation using a single deep deterministic neural network. In *International Conference on Machine Learning*, pp. 9690–9700. PMLR, 2020. 49, 50
- Van De Meent, J.-W., Paige, B., Yang, H., and Wood, F. An introduction to probabilistic programming. *arXiv preprint arXiv:1809.10756*, 2018. 187
- Van Der Maaten, L., Postma, E., and Van den Herik, J. Dimensionality reduction: a comparative. *Journal of Machine Learning Research*, 10(66-71):13, 2009. 78
- Vapnik, V. N. and Chervonenkis, A. Y. On the uniform convergence of relative frequencies of events to their probabilities. In *Measures of complexity*, pp. 11–30. Springer, 1968. 63
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. In *Advances in Neural Information Processing Systems*, NIPS’17, pp. 6000–6010, Red Hook, NY, USA, 2017. Curran Associates Inc. 21
- Vedaldi, A. and Soatto, S. Quick shift and kernel methods for mode seeking. In *European Conference on Computer Vision*, pp. 705–718. Springer, 2008. 123
- Vempati, A. S., Khurana, H., Kabelka, V., Flueckiger, S., Siegwart, R., and Beardsley, P. A virtual reality interface for an autonomous spray painting uav. *IEEE Robotics and Automation Letters*, 4(3):2870–2877, July 2019. 130

- Vey, S., Lang, H., Nayeri, C., Paschereit, C. O., and Pechlivanoglou, G. Extracting quantitative data from tuft flow visualizations on utility scale wind turbines. In *Journal of Physics: Conference Series*, number 524-1, pp. 012011. IOP Publishing, 2014. 113, 114, 115, 116
- Wagner, D. and Schmalstieg, D. Artoolkitplus for pose tracking on mobile devices. In *Proceedings of 12th Computer Vision Winter Workshop*, pp. 139–146, 01 2007. 128, 131, 137, 138, 139, 148
- Wang, H., Shi, X., and Yeung, D.-Y. Natural-parameter networks: A class of probabilistic neural networks. *Advances in Neural Information Processing Systems*, 29, 2016. 49
- Wang, H., Fu, T., Du, Y., Gao, W., Huang, K., Liu, Z., Chandak, P., Liu, S., Van Katwyk, P., Deac, A., et al. Scientific discovery in the age of artificial intelligence. *Nature*, 620 (7972):47–60, 2023a. 21, 22
- Wang, J. and Olson, E. Apriltag 2: Efficient and robust fiducial detection. In *International Conference on Intelligent Robots and Systems*, pp. 4193–4198, 2016. 131, 138, 148, 168, 184
- Wang, L., Zhang, X., Su, H., and Zhu, J. A comprehensive survey of continual learning: theory, method and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024. 26, 100
- Wang, T. S., Marton, Z.-C., Brucker, M., and Triebel, R. How robots learn to classify new objects trained from small data sets. In *Conference on Robot Learning*, pp. 408–417, 2017. 62
- Wang, X., Jin, Y., Schmitt, S., and Olhofer, M. Recent advances in bayesian optimization. *ACM Computing Surveys*, 55(13s):1–36, 2023b. 23
- Wang, Y. and Solomon, J. M. Deep closest point: Learning representations for point cloud registration. In *International Conference on Computer Vision*, pp. 3523–3532, 2019. 131
- Wang, Z., Ku, A., Baldridge, J., Griffiths, T., and Kim, B. Gaussian process probes (gpp) for uncertainty-aware probing. *Advances in Neural Information Processing Systems*, 36: 63573–63594, 2023c. 49
- Wei, Y. and Ji, S. Scribble-based weakly supervised deep learning for road surface extraction from remote sensing images. *IEEE Transactions on Geoscience and Remote Sensing*, 60: 1–12, 2021. 121
- Welling, M. and Teh, Y. W. Bayesian learning via stochastic gradient langevin dynamics. In *International Conference on Machine Learning*, pp. 681–688. PMLR, 2011. 48, 50
- Wen, Y., Tran, D., and Ba, J. Batchensemble: an alternative approach to efficient ensemble and lifelong learning. In *International Conference on Learning Representations*, 2020. 49, 63
- Wenger, J., Kjellström, H., and Triebel, R. Non-parametric calibration for classification. In *International Conference on Artificial Intelligence and Statistics*, Proceedings of Machine Learning Research, 2020. 78, 89, 253

- Wenzel, F., Roth, K., Veeling, B., Swiatkowski, J., Tran, L., Mandt, S., Snoek, J., Salimans, T., Jenatton, R., and Nowozin, S. How good is the bayes posterior in deep neural networks really? In *International Conference on Machine Learning*, volume 119, pp. 10248–10259. PMLR, 2020. 46, 48, 55, 56, 58, 59, 62, 64, 65, 71, 188
- Whitney, D., Rosen, E., Phillips, E., Konidaris, G., and Tellex, S. Comparing robot grasping teleoperation across desktop and virtual reality with ros reality. *Robotics Research*, pp. 335–350, 2020. 128
- Wieser, D., Bonitz, S., Lofdahl, L., Broniewicz, A., Nayeri, C., Paschereit, C., and Larsson, L. Surface flow visualization on a full-scale passenger car with quantitative tuft image processing. Technical report, SAE Technical Paper, 2016. 113, 115, 116
- Williams, P. M. Bayesian regularization and pruning using a laplace prior. *Neural Computation*, 7(1):117–143, 1995. 46
- Wilson, A. G. The case for bayesian deep learning. *arXiv preprint arXiv:2001.10995*, 2020. 45
- Wilson, A. G. and Izmailov, P. Bayesian deep learning and a probabilistic perspective of generalization. *Advances in Neural Information Processing Systems*, 33:4697–4708, 2020. 26, 45, 60
- Wilson, A. G., Hu, Z., Salakhutdinov, R., and Xing, E. P. Deep kernel learning. In *Artificial intelligence and statistics*, pp. 370–378. PMLR, 2016. 46, 87
- Wilson, A. G., Izmailov, P., Hoffman, M. D., Gal, Y., Li, Y., Pradier, M. F., Vikram, S., Foong, A., Lotfi, S., and Farquhar, S. Evaluating approximate inference in bayesian deep learning. In *NeurIPS 2021 Competitions and Demonstrations Track*, pp. 113–124. PMLR, 2022. 188, 194
- Wilson, J., van der Heide, C., Hodgkinson, L., and Roosta, F. Uncertainty quantification with the empirical neural tangent kernel. *arXiv preprint arXiv:2502.02870*, 2025. 194
- Wlach, S., Schwarzbach, M., and Laiacker, M. Dlr hableg–high altitude balloon launched experimental glider. In *22nd ESA Symposium on European Rocket and Balloon Programmes and Related Research*, pp. 385–392. ESA Communications, 2015. 116
- Wold, S., Esbensen, K., and Geladi, P. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987. 78
- Wolf, M. Mathematical foundations of supervised learning, 2018. 201
- Wong, K., Wang, S., Ren, M., Liang, M., and Urtasun, R. Identifying unknown instances for autonomous driving. In *Conference on Robot Learning*, volume 100, pp. 384–393. PMLR, 30 Oct–01 Nov 2020. 92
- Wonsick, M. and Padir, T. A systematic review of virtual reality interfaces for controlling and interacting with robots. *Applied Sciences*, 10(24):9051, 2020. 130
- Wu, A., Nowozin, S., Meeds, E., Turner, R. E., Hernández-Lobato, J. M., and Gaunt, A. L. Deterministic variational inference for robust bayesian neural networks. *International Conference on Learning Representations*, 2019. 47, 50, 63, 69

- Wu, Y., Song, J., Sun, J., Zhu, F., and Chen, H. Aerial grasping based on vr perception and haptic control. In *2018 IEEE International Conference on Real-time Computing and Robotics*, pp. 556–562, 2018. 127
- Xiangdong, X., Bo, L., and Jiannan, G. Asset management of oil and gas pipeline system based on digital twin. *IFAC-PapersOnLine*, 2020. 168
- Yashin, G. A., Trinitatova, D., Agishev, R. T., Ibrahimov, R., and Tsetserukou, D. Aerovr: Virtual reality-based teleoperation with tactile feedback for aerial manipulation. In *2019 19th International Conference on Advanced Robotics*, pp. 767–772, 2019. 127, 130
- Yow, J.-A., Garg, N. P., and Ang, W. T. Shared autonomy of a robotic manipulator for grasping under human intent uncertainty using pomdps. *IEEE Transactions on Robotics*, 2023. 165
- Yu, H., Hartmann, M., Sanchez, B. W. M., Girolami, M., and Klami, A. Riemannian laplace approximation with the fisher metric. In *International Conference on Artificial Intelligence and Statistics*, pp. 820–828. PMLR, 2024. 48
- Yu, T., Yang, Y., Li, D., Hospedales, T., and Xiang, T. Simple and effective stochastic neural networks. In *AAAI Conference on Artificial Intelligence*, number 4, pp. 3252–3260, 2021. 49
- Zenke, F., Poole, B., and Ganguli, S. Continual learning through synaptic intelligence. In *International Conference on Machine Learning*, volume 70, pp. 3987–3995. PMLR, 06–11 Aug 2017. 78
- Zhang, G., Sun, S., Duvenaud, D., and Grosse, R. Noisy natural gradient as variational inference. In *International Conference on Machine Learning*, pp. 5852–5861. PMLR, 2018. 50, 69
- Zhang, J. and Singh, S. Low-drift and real-time lidar odometry and mapping. *Autonomous Robots*, 41(2):401â€“416, feb 2017. 128, 143
- Zhang, J., Yu, X., Li, A., Song, P., Liu, B., and Dai, Y. Weakly-supervised salient object detection via scribble annotations. In *Conference on Computer Vision and Pattern Recognition*, pp. 12546–12555, 2020a. 116
- Zhang, Z., Dai, Y., and Sun, J. Deep learning based point cloud registration: an overview. *Virtual Reality & Intelligent Hardware*, 2(3):222–246, 2020b. 131
- Zhang, Z., Zhu, Y., and Zhu, S.-C. Graph-based hierarchical knowledge representation for robot task transfer from virtual to physical world. In *International Conference on Intelligent Robots and Systems*, pp. 11139–11145, 2020c. 130
- Zhao, M., Okada, K., and Inaba, M. Versatile articulated aerial robot dragon: Aerial manipulation and grasping by vectorable thrust control. *International Journal of Robotics Research*, 2023. 165
- Zhao, M. D., Simmons, R., Admoni, H., and Bajcsy, A. Conformalized Teleoperation: Confidently Mapping Human Inputs to High-Dimensional Robot Actions. In *Robotics: Science and Systems*, July 2024. 165

- Zhou, L., Zhang, C., and Wu, M. D-linknet: Linknet with pretrained encoder and dilated convolution for high resolution satellite imagery road extraction. In *Conference on Computer Vision and Pattern Recognition Workshops*, pp. 182–186, 2018. 116, 120
- Zhou, Q.-Y., Park, J., and Koltun, V. Fast global registration. In *European Conference on Computer Vision*, pp. 766–782, 2016. 131, 141, 150
- Zitkovich, B., Yu, T., Xu, S., Xu, P., Xiao, T., Xia, F., Wu, J., Wohlhart, P., Welker, S., Wahid, A., et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning*, pp. 2165–2183. PMLR, 2023. 194