Karlsruher Institut für Technologie

Autonomous Learning Robots Lab

Probabilistic World and Behavior Models for Embodied Agents

Zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

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

genehmigte

Dissertation

von

Philipp Becker

geb. in Groß-Umstadt

Tag der mündlichen Prüfung: 21. Oktober 2025
1. Referent: Prof. Dr. Gerhard Neumann
2. Referent: Prof. Dr. Sebastian Trimpe

Abstract

Embodied agents must operate safely and efficiently in complex, dynamic, and unstructured environments. In these settings, they face the inherent uncertainty, unpredictability, and ambiguity of the world and other agents within it, particularly humans. The agents' limited perception further amplifies these challenges, resulting in noisy and partial observations that may contain distractions. Consequently, simple deterministic models are an insufficient basis for decision-making and can lead to inefficient or even dangerous behavior.

Probabilistic models provide a principled and powerful framework for representing knowledge, accounting for ambiguities, and quantifying uncertainties. Combined with modern deep learning techniques, they facilitate learning probabilistic world and behavior models directly from highly complex data. Additionally, imposing structure through latent variables and graphical models can guide these models to learn meaningful representations that support informed decision-making. However, learning such structured, uncertainty-aware models presents its own set of challenges, especially within the context of learning behavior through Imitation or Reinforcement Learning (RL).

This thesis addresses three such challenges by developing novel probabilistic models, training approaches, and inference schemes. We first tackle the problem of learning from diverse human demonstrations by introducing a learning approach based on the Information Projection. Unlike standard methods that can average over distinct behaviors, which can lead to unsafe actions, our approach enables learning mixture models that represent multimodal human behavior more faithfully. Next, we address the challenge of learning world models for model-based RL with deep State Space Models, which often fail to represent different sources of uncertainty appropriately. After analyzing the shortcomings of prior approaches, we propose a more principled alternative that leverages the foundations of Kalman smoothing for inference. This enables our model to explicitly distinguish between environmental uncertainty and model uncertainty, resulting in more robust decision-making under partial observability. Finally, we employ similar state space approaches to learn multi-sensor representations by developing a unified framework that combines reconstruction-based and contrastive learning. Our framework allows tailoring the loss to the properties of the respective sensors and learning concise, task-relevant representations for RL.

Together, these contributions provide new tools and deeper insights for creating capable and reliable embodied agents. By advancing the theory and practice of probabilistic world and behavior models, this thesis contributes to the development of intelligent systems that can better navigate the uncertainties and complexities of the world.

Zusammenfassung

Embodied Agents müssen in komplexen, dynamischen und unstrukturierten Umgebungen sicher und effizient agieren. Dort sie sind mit der inhärenten Unsicherheit, Unvorhersehbarkeit und Mehrdeutigkeit der Welt sowie anderer Agenten, insbesondere Menschen, konfrontiert. Ihre eingeschränkte Wahrnehmung verstärkt diese Herausforderungen zusätzlich, da sie zu verrauschten und unvollständigen Beobachtungen führt, die auch ablenkende Elemente enthalten können. Einfache deterministische Modelle bieten daher keine ausreichende Grundlage für die Entscheidungsfindung und können ineffizientes oder sogar gefährliches Verhalten zur Folge haben.

Probabilistische Modelle bilden ein fundiertes und leistungsstarkes Framework, um Wissen zu repräsentieren, Mehrdeutigkeiten zu berücksichtigen und Unsicherheiten zu quantifizieren. In Verbindung mit modernen Deep-Learning-Techniken ermöglichen sie es, probabilistische Welt- und Verhaltensmodelle direkt aus komplexen Daten zu erlernen. Mithilfe latenter Variablen und grafischer Modelle lassen sich diese Modelle zudem so strukturieren, dass sie sinnvolle Repräsentationen entwickeln, die eine fundierte Entscheidungsfindung unterstützen. Das Erlernen solcher strukturierter, unsicherheitsbewusster Modelle bringt jedoch erhebliche Herausforderungen mit sich, insbesondere im Kontext des Erlernens von Verhalten mittels Imitation- oder Reinforcement Learning (RL).

Diese Arbeit adressiert drei dieser Herausforderungen durch die Entwicklung neuartiger probabilistischer Modelle, Trainingsansätze und Inferenzverfahren. Zunächst stellen wir einen auf Information Projection basierenden Lernansatz vor, um aus multimodalen menschlichen Demonstrationen zu lernen. Im Gegensatz zu Standardmethoden, die über verschiedene Verhaltensweisen mitteln und dadurch unsicheres Handeln riskieren, erlaubt unser Ansatz das Erlernen von Mixture Models, die multimodales menschliches Verhalten präzise abbilden. Anschließend befassen wir uns mit dem Problem, dass Weltmodelle für model-based RL, welche als Deep State Space Models realisiert werden, häufig verschiedene Quellen der Unsicherheit nicht angemessen darstellen können. Nach einer Analyse der Schwächen bestehender Ansätze schlagen wir eine Alternative vor, die die Inferenz auf den Grundlagen des Kalman Filters aufbaut. Dadurch kann unser Modell explizit zwischen Umweltunsicherheit und Modellunsicherheit unterscheiden und unterstützt eine robustere Entscheidungsfindung bei partieller Beobachtbarkeit. Schließlich kombinieren wir beim Erlernen von Multisensorrepräsentationen rekonstruktionsbasiertes und kontrastives Lernen in einem einheitlichen Framework. So kann das Lernverfahren an die jeweiligen Sensoreigenschaften angepasst werden, um prägnante und aufgabenrelevante Repräsentationen für RL zu erlernen.

Diese Beiträge liefern neue Werkzeuge und tiefere Einsichten für die Entwicklung zuverlässiger und leistungsfähiger Embodied Agents. Die Arbeit leistet damit einen Beitrag zur Weiterentwicklung der Theorie und Praxis probabilistischer Welt- und Verhaltensmodelle und zur Schaffung intelligenter Systeme, die besser mit den Unwägbarkeiten und der Komplexität der Welt umgehen können.

Acknowledgments

First, I want to express my gratitude to Geri, whose guidance, expertise, and patience have been invaluable throughout my PhD journey. Your belief in my work and skills, which was often stronger than my own, as well as your optimism, have been a constant source of motivation and a key reason for my success. Thank you for your feedback, for always listening to my ideas and rants, as well as allowing me to take the opportunities I wanted to take and pass on those I didn't.

This gratitude extends to all my colleagues and friends at the Autonomous Learning Robots Lab, who have been a source of inspiration, support, and camaraderie throughout my PhD journey. You made night shifts and crunching deadlines bearable, sometimes even enjoyable, and I am grateful for the many discussions, shared moments, and humor that have enriched my experience. Special gratitude goes to Onur for working with me almost from the very beginning, at Bosch, the KIT, and the FZI, for being a great collaborator and friend, and reminding me more than once that I probably need to stop overthinking (and the Raki). To Niklas for his friendship and for teaching me a thing or two about myself, organizing things, volcanoes, and islands, as well as the pleasures of climbing plastic rocks in a gym with a bunch of sweaty people. To Max, for always being there to offer advice on research, life, how to be kinder, and Doppelkopf. To Philipp for being open to discussing the details of filtering with Kalman filters and the many invitations to the Casa Vogt. To Balazs for always telling me when I'm wrong, even when I'm not, and for our many discussions, debates, and climbing sessions. To Fabian and Vaisakh for being great colleagues and collaborators, as well as for having great career advice. To Nic for always helping out, whether you need your bike fixed, something 3D printed, or something else. To Aleks for providing the best feedback, in particular also for this thesis. To Christine for relentlessly resolving any and all possible and impossible administrative issues, as well as helping me to stay sane in the madhouse that is public service bureaucracy. To Tai, Michael, Andy, Vien, Bruce, and Denis for various research collaborations, proofreading, discussions, and feedback. To all my students and collaborators whose contributions, insights, and often unique perspectives have greatly enriched my research.

I would also like my family and friends outside of the lab for their support and for constantly reminding me that there is much more to life than work and research. Here, I would like to extend my deepest gratitude to my parents for their unwavering love and encouragement throughout my life and academic journey. The knowledge that I can always count on your unconditional and unlimited support has been a constant source of strength for me. You made me who I am today and gave me the tools and freedom to pursue my dreams.

Lastly, I would like to thank Prof. Sebastian Trimpe and TT-Prof. Peer Nowack for serving as referee and examiner for this thesis.

Contents

Abstract . . . . **i**

Zusammenfassung . . . . **iii**

List of Figures . . . . **xi**

List of Tables . . . . **xv**

1. Introduction **1**
- 1.1. Overview of Thesis Contributions and Structure 3

2. Foundations **9**
- 2.1. Probabilistic Modeling and Inference 9
- 2.2. Variational Inference and Learning in Latent Variable Models 16
- 2.3. Reinforcement Learning for Behavior Learning with Probabilistic World Models 24
- 2.4. Distribution Matching to Learn Behavior by Imitation 34

3. Expected Information Maximization: Using the I-Projection for Mixture Density Estimation **41**
- 3.1. Introduction 42
- 3.2. Related Work 43
- 3.3. Expected Information Maximization 44
- 3.4. Qualitative Comparison to Related Generative Approaches 47
- 3.5. Empirical Evaluation 49
- 3.6. Conclusion 53
- 3.7. Follow Ups 54

4. Recurrent Kalman Networks: Factorized Inference in High-Dimensional Deep Feature Spaces **61**
- 4.1. Introduction 62
- 4.2. Factorized Inference in Deep Feature Spaces 65
- 4.3. Evaluation and Experiments 69
- 4.4. Conclusion 75
- 4.5. Follow Ups 76

5. On Uncertainty in Deep State Space Models for Model-Based Reinforcement Learning **83**
- 5.1. Introduction 84

5.2. Inference and Learning in State Space Models 86
5.3. Variational Recurrent Kalman Networks 92
5.4. Evaluation 96
5.5. Related Work 102
5.6. Conclusion 104

6. KalMamba: Towards Efficient Probabilistic State Space Models for RL under Uncertainty 105
6.1. Introduction 105
6.2. Related Work 107
6.3. State Space Models for Reinforcement Learning 109
6.4. KalMamba 110
6.5. Experiments 113
6.6. Conclusion 116

7. Combining Reconstruction and Contrastive Methods for Multi-Sensor Representations in RL 119
7.1. Introduction 120
7.2. Related Work 121
7.3. Combining Contrastive Approaches and Reconstruction for State Space Representations 122
7.4. Experiment Setup 126
7.5. Discussion of Results 131
7.6. Conclusion 135

8. Conclusion 137
8.1. Future Work and Outlook 139

Bibliography 141

List of Publications 155

List of Supervised Student Theses 159

A. Appendix for Chapter 3 161
A.1. Derivations 161
A.2. Hyperparameters 164
A.3. Further Discussion on Prior Works 165

B. Appendix for Chapter 4 167
B.1. Simplified Kalman Filter Formulas 167
B.2. Root Mean Square Error Results 170
B.3. Visualization of Imputation Results 170
B.4. Network Architectures and Hyper Parameters 170
B.5. Observation Noise generation process 174

C. Appendix for Chapter 5 . . . . **175**
C.1. Derivations 175
C.2. Details on Experiment Setup, Baselines and Hyperparameters 178
C.3. Comprehensive Experiment Results and Visualizations 181

D. Appendix for Chapter 6 . . . . **195**
D.1. Hyperparameters and Implementation Details 195
D.2. Additional Results 195

E. Appendix for Chapter 7 . . . . **201**
E.1. Architecture Details and Training 201
E.2. Details on Baselines and Ablations. 203
E.3. Complete Results 206

List of Figures

2.1. Graphical Model of a State Space Model . . . 13
2.2. Overview of a Variational Autoencoding for Images . . . 19
2.3. Overview of State Space Models as World Models for RL . . . 33
2.4. Maximum Likelihood vs. I-Projection . . . 35

3.1. Illustration of the I-Projection vs. the M-Projection for modeling behavior . 43
3.2. Illustrative example of the benefits of EIM versus an adversarial formulation 48
3.3. EIM Ablation Results . . . 50
3.4. EIM Line Reacher Results . . . 50
3.5. EIM Pedestrian and Traffic Prediction Results . . . 51
3.6. Samples from Pedestrian Prediction Experiments . . . 52
3.7. EIM Obstacle Avoidance Results . . . 52
3.8. Example of Behavior Learned on a Real Robot by VIGOR . . . 56

4.1. Overview of the RKN architecture . . . 66
4.2. Quality of Uncertainty Predictions of RKN vs. LSTM . . . 72
4.3. Predictions of RKN on Multiple Pendulum . . . 73
4.4. RKN Predictions on Pneumatic Brook Robot Arm . . . 74
4.5. Images of the Robots Used to Evaluate the ac-RKN . . . 77
4.6. Example Trajectories of the SRKN's Predictions of Multimodal Human Driving Behavior . . . 82

5.1. Graphical Models of SSM and RSSM . . . 86
5.2. Illustrative Example of the RSSMs Overestimation of Aleatoric Uncertainty . 90
5.3. Overview of the VRKN's Architecture and Dynamics Model . . . 92
5.4. Comparison of the RSSM and Naive Smoothing Apporaches . . . 97
5.5. Comparison of VRKN and RSSM on Standard Tasks . . . 98
5.6. Examples of Occlusion Tasks . . . 99
5.7. Comparison of VRKN and RSSM on Tasks with Occlusions . . . 100
5.8. Qualitative Comparison of Image Reconstructions from VRKN and RSSM . . 100
5.9. Comparison of VRKN and RSSM on Tasks with Missing Observations . . . 101
5.10. Comparison of VRKN and RSSM on Sensor Fusion Tasks . . . 102

6.1. Overview of the KalMamba Model . . . 106
6.2. Schematic of KalMamba's Mamba-based dynamics backbone . . . 111
6.3. Results and Ablations of KalMamba on Image Tasks . . . 113
6.4. Results of KalMamba on State-Based Noisy Tasks . . . 114
6.5. KalMamba Wall-Clock Time Evaluations . . . 115

6.6. KalMamba, RSSM, and VRKN Runtime Comparisons 116

7.1. Overview of Contrastive Reconstructive Aggregated representation Learning 121
7.2. Example Images of Modified Deep Mind Control Suites 128
7.3. Example Images of the Environments in the Locomotion Suite 129
7.4. Example Images of the Environments in the Manipulation Suite 129
7.5. Results on tasks with Video Backgrounds 131
7.6. Results on tasks with Occlusions 132
7.7. Qualitative Analysis of Learned Representations 133
7.8. Results for Locomotion Suite 133
7.9. Results for Manipulation Suite 134

B.1. Visualizations of Pendulum and Quad Link Imputation Tasks 172
B.2. Example Images for the Multiple Pendulum Experiments 173

C.1. Overview of MCD-RSSM Architecture 179
C.2. Reward Curves of RSSM vs Baselines for Planet Agents 182
C.3. Box Plots of RSSM vs Baselines for Planet Agents 183
C.4. Reward Curves of RSSM vs Baselines for Dreamer Agents 183
C.5. Box Plots of RSSM vs Baselines for Dreamer Agents 184
C.6. Reward Curves of VRKN vs RSSM for Planet Agents 185
C.7. Box Plots of VRKN vs RSSM for Planet Agents 185
C.8. Reward Curves of VRKN vs RSSM for Dreamer Agents 186
C.9. Box Plots of VRKN vs RSSM for Planet Agents 187
C.10. Reward Curves for all Agents on Tasks with Disc Occlusions 187
C.11. Box Plots for all Agents on Tasks with Disc Occlusions 188
C.12. Reward Curves for all Agents on Tasks with Wall Occlusions 188
C.13. Box Plots for all Agents on Tasks with Wall Occlusions 189
C.14. Reward Curves for all Agents on Tasks with Missing Information 189
C.15. Box Plots for all Agents on Tasks with Missing Information 189
C.16. Reward Curves for all Agents on Tasks with Missing Observations and Transition Noise 190
C.17. Box Plots for all Agents on Tasks with Missing Observations and Transition Noise 190
C.18. Reward Curves for all Agents on Sensor Fusion Tasks 191
C.19. Box Plots for all Agents on Sensor Fusion Tasks 191
C.20. Reward Curves for all Agents on Sensor Fusion Tasks with Transition Noise 192
C.21. Box Plots for all Agents on Sensor Fusion Tasks with Transition Noise 192
C.22. Trajectory Samples of Cup Catch Task with Disc Occlusions 192
C.23. Trajectory Samples of the Cheetah Run Task with Disc Occlusions 193
C.24. Trajectory Samples of the Cartpole Swingup Task with Disc Occlusions . . . 193
C.25. Trajectory Samples of the Walker Walk Task with Disc Occlusions 193
C.26. Samples for Initial State of Cup Catch Tasks with Disc Occlusions 194

D.1. Per Environment Evaluations of the Image-Based Tasks 197
D.2. Per Environment Evaluations of the KalMamba Evaluations 198

D.3. Per Environment Evaluations of the Noisy State-Based Tasks 199

E.1. Aggregated Rewards for Model-Free Agents on the DeepMind Control Suite 207
E.2. Aggregated Rewards for Model-Based Agents on the DeepMind Control Suite 208
E.3. Performance Profiles for Model-Free Agents on the DeepMind Control Suite 209
E.4. Performance Profiles for Model-Based Agents on the DeepMind Control Suite 210
E.5. Aggregated Results and Performance Profiles for the Locomotion Suite . . . 211
E.6. Aggregated Results and Performance Profiles for the Manipulation Suite . . 212
E.7. Per Environment Results for Model-Free Agents with Standard Images . . . 213
E.8. Per Environment Results for Model-Free Agents with Video Background . . 214
E.9. Per Environment Results for Model-Free Agents with Occlusions 215
E.10. Per Environment Results for Model-Based Agents with Standard Images . . 216
E.11. Per Environment Results for Model-Based Agents with Video Background . 217
E.12. Per Environment Results for Model-Based Agents with Occlusions 218
E.13. Per Environment Results for the Locomotion Suite. 219
E.14. Per Environment Results for the Manipulation Suite. 220

List of Tables

4.1. Qualitative Comparison of the RKN with Related Work. 64
4.2. RKN Pendulum State Estimation Results 70
4.3. RKN Pendulum Image Imputation Results 71
4.4. RKN Multiple Pendulum State Estimation Results 72
4.5. RKN Quad Link State Estimation Results 73
4.6. RKN Results on KITTI Dataset 74

6.1. Comparing the inference models and capabilities for smoothing (Smooth) and time-parallel (Parallel) execution of recent SSMs for RL. 109

7.1. Proprioception / Non-Proprioception Splits for the DeepMind Control Suite 127
7.2. Proprioception / Non-Proprioception Splits for the Locomotion Suite 128

B.1. RMSE Results 171

C.1. Splits for Proprioception and Non-Proprioception for VRKN Fusion Experiments 180
C.2. Final Performance of RSSM vs Baselines for Planet Agents 182
C.3. Final Performance of RSSM vs Baselines for Dreamer Agents 184
C.4. Final Performance of VRKN vs RSSM for Planet Agents 186
C.5. Final Performance of VRKN vs RSSM for Planet Agents 187
C.6. Final Performance of all Agents on Tasks with Disc Occlusions 188
C.7. Final Performance of all Agents on Tasks with Wall Occlusions 188
C.8. Final Performance of all Agents on Tasks with Missing Observations 190
C.9. Final Performance of all Agents on Tasks with Missing Observations and Transition Noise 190
C.10. Final Performance of all Agents on Sensor Fusion Tasks 191
C.11. Final Performance of all Agents on Sensor Fusion Tasks with Transition Noise 191

D.1. World Model Hyperparameters for KalMamba Experiments 196
D.2. SAC Hyperparameters for KalMamba Experiments 197

E.1. Hyperparameters for Model-Free Policy Learning with the Soft Actor-Critic. 203
E.2. Hyperparameters for Model-Based Policy Learning with Latent Imagination. 204

1. Introduction

In our increasingly automated world, embodied agents such as robots or autonomous vehicles need to operate in complex and dynamic environments. To act safely and efficiently, these agents must accurately represent and understand their surroundings, the consequences of their actions, and the behavior of other agents, such as humans. However, real-world conditions and human behaviors are inherently uncertain, unpredictable, and ambiguous. Adding to this complexity, the agent's perception is often limited, with observations that are noisy, partial, and may contain distracting details or irrelevant information. Consequently, simple deterministic models of the world and behaviors are insufficient and can lead to inefficient or even dangerous actions. Instead, agents require models that can robustly represent these uncertainties, account for ambiguities, and are aware of what they do not know.

Probabilistic models provide a principled framework for representing, quantifying, and reasoning about uncertainties (Murphy, 2012). Combined with modern deep learning techniques (Goodfellow et al., 2016), they enable the learning of complex probabilistic representations from high-dimensional data. Furthermore, by imposing structure on these models using latent variables and graphical models, we can guide them to learn meaningful representations. For example, such models can help disentangle underlying causes from raw sensory observations, naturally capture multimodality (Bishop, 2006), and introduce information bottlenecks (Kingma and Welling, 2013; Alemi et al., 2016), which help focus on relevant aspects. However, the need for awareness of uncertainty and added structure makes learning such models more challenging, necessitating specialized learning methods.

Additionally, when used for embodied agents, we often ultimately care about learning behavior, and we must consider probabilistic models in this context. In modern machine learning, the two primary paradigms for learning behavior are Imitation Learning (IL) (Pomerleau, 1991; Osa et al., 2018) and Reinforcement Learning (RL) (Sutton and Barto, 2018). In IL, the agent learns to copy the behavior of an expert, typically a human, who demonstrates the desired behavior. While this approach provides a direct way to learn behavior, the quality and scope of the demonstrations fundamentally limit the agent's performance. In contrast, RL enables an agent to learn through trial-and-error interactions with its environment, optimizing actions to maximize a cumulative reward signal. This approach enables the discovery of novel behavior, unlocks skills for which there are no experts (Degrave et al., 2022), and can surpass the best humans (Silver et al., 2016, 2018). However, it often requires extensive and sometimes unsafe exploration. Which paradigm is more suitable often depends on the task and hand.

Yet, crucially, both paradigms introduce additional challenges when building and learning probabilistic models. In IL, the data is often collected by other agents, such as humans, and in RL the agent collects its own data through exploration. Both settings violate many of the assumptions underlying common supervised learning (Ross et al., 2011; Mnih et al., 2015). Additionally, when used for behavior learning we often care about the quality, safety, and efficiency of the resulting behavior, which is not necessarily captured by the model's training objective (Lambert et al., 2020). If not carefully mitigated, this mismatch or other conflicting objectives and quality measures can lead to suboptimal or even dangerous results. In this thesis, we focus on three such challenges, which we consider particularly relevant for embodied agents.

The first challenge arises when embodied agents need to predict or learn from human behavior. Humans often demonstrate remarkable versatility in their actions. When faced with the same task, different humans often employ different strategies, and even the same person may not use the same strategy when solving the task a second time. While this versatility is one of the main strengths of humans, it can pose considerable challenges when learning from human demonstrations. Therefore, an agent learning from human demonstrations should be able to effectively represent this multimodal behavior. Crucially, it should avoid averaging over different modes of behavior, as this can lead to disastrous actions. For instance, if humans avoid an obstacle by passing it on either the left or right side, averaging would lead to running into the obstacle head-on. While probabilistic mixture models can capture multimodality, common training objectives, such as maximum likelihood, inherently encourage mode averaging.

The second challenge arises as embodied agents must learn to form effective representations of the world itself to make safe and efficient decisions. Here, the agent typically forms these representations from high-dimensional, noisy, partial, and ambiguous observations due to occlusions or a limited field of view. One efficient way to learn behavior with such representation using RL is by predicting the future state of the world, given potential actions (Chua et al., 2018; Janner et al., 2019). This prediction ability allows agents to plan their actions more effectively by simulating different scenarios and evaluating their outcomes before execution. Again, uncertainty is a crucial factor as assuming the current representation is correct and the prediction is flawless would lead to overconfident and potentially dangerous decisions. However, if the model is aware of its uncertainties, it allows the agent to make more informed decisions by reasoning about all states of the world it might be in and all possible outcomes of its actions. Especially during learning, it is crucial that the models can disentangle the source of the uncertainties, particularly whether they are due to limited perception, a lack of experience, or insufficient data. This raises the question of how we can learn probabilistic world models that effectively represent the world and its uncertainties, disentangle them, and are amenable to informed decision-making.

The third challenge arises as most embodied agents perceive the world through multiple sensors with different properties. Similar to humans, embodied agents typically have a precise understanding of their internal state through proprioception but must rely on less precise sensors, like vision, to comprehend their surroundings. These sensors often provide high-dimensional, noisy, and unfiltered signals that can contain irrelevant details, which distract from decision-making and lead to inefficient representations. In humans, the brain has evolved to unconsciously filter irrelevant information from raw sensory input. It also integrates information from past observations where necessary to form a concise representation of the current state of the world for the task at hand. This ability to prioritize and condense information is crucial for effective decision-making and action planning in complex environments. It raises the question of how we can design world models and training objectives that equip embodied agents with such capabilities.

In this thesis, we address these challenges by developing novel probabilistic models, learning objectives, and inference schemes. We introduce new algorithms that leverage structured latent variable models to learn meaningful representations from high-dimensional, noisy, and multimodal data. Our contributions include new training methods designed to avoid averaging when learning from multimodal human demonstrations, principled and efficient model and inference schemes for modeling and disentangling uncertainties in deep State Space Models (SSMs), and a unified framework for learning from multiple sensors with different properties.

1.1. Overview of Thesis Contributions and Structure

We condense the challenges outlined above into three distinct research questions, which we will answer in this thesis.

Q1 How can we learn multimodal probabilistic models that focus on modes they can represent and avoid averaging? (Chapter 3)

Q2 How can we build a computationally efficient world model for RL that captures and propagates uncertainties in the world while separating them from uncertainties due to lack of training data? (Chapter 5 and Chapter 6)

Q3 How can we use probabilistic world models to combine information from multiple sensors with highly different properties into a single, concise representation as a basis for RL? (Chapter 7)

Before taking a closer look at how we answer these questions, Chapter 2 reviews the probabilistic modeling techniques and learning approaches used throughout this work. Additionally, it provides background on Imitation and Reinforcement Learning.

In Chapter 3, we address the first question by introducing a novel method for distribution matching for latent variable models, specifically mixture models. This method employs an alternative objective, namely the Information Projection, which enables focusing on modes that the model can represent. This objective is well-established in Variational Inference (Jordan et al., 1999; Arenz et al., 2018), and we adapt it for IL by providing efficient algorithms to optimize it based on demonstration data.

To address the second and third questions, we focus on SSMs to learn effective probabilistic world models from challenging observations. These naturally support awareness of uncertainty and the capability to integrate information from multiple sensors. SSMs are well-established probabilistic models for dynamical systems and ubiquitous in control and robotics. When combined with deep learning, they are particularly well suited for learning representations from sequences of high-dimensional observations. Additionally, as we now aim to learn targeted, goal-driven behavior, given the sensory limitations of the embodied agent itself, we can no longer rely on imitation but must consider Reinforcement Learning (RL) based approaches, which enable agents to learn from trial-and-error interactions with the environment. However, this paradigm introduces additional challenges in model learning, as the agent must incrementally collect its own data for improvement.

Before addressing the second question, Chapter 4 recapitulates our prior work on efficient inference in high-dimensional feature spaces, which combines Kalman filtering and deep learning. It lays the foundation for the state space approaches to world modeling introduced in the following chapters. In Chapter 5, we start addressing the second question by analyzing the limitations of existing SSM approaches to world modeling (Hafner et al., 2019, 2020), particularly with respect to representing and disentangling uncertainties. Here, we improve on this uncertainty handling by building on the foundations of Kalman filtering (Kalman, 1960) and smoothing (Rauch et al., 1965), as well as the methods introduced in Chapter 4. While effective, the resulting approach lacks computational efficiency, which we address in Chapter 6 by presenting a unified approach combining probabilistic SSMs with recent advances in structured SSMs for sequence modeling (Smith et al., 2022; Gu and Dao, 2023).

Maintaining SSMs as our model and RL as the paradigm for behavior learning of choice, Chapter 7 addresses the third question regarding multi-sensor representations. For this setting, we propose a unified framework to learn multi-sensor representations from multiple sensors. To account for the different properties and characteristics of the sensors, we propose a principled approach that combines reconstruction and contrastive learning (Oord et al., 2018) to learn effective representations for RL.

Finally, Chapter 8 concludes the thesis, summarizes its contributions, and explores potential future research directions.

Below, we provide summaries of the four individual papers which form the main contributions of this thesis.

The following four sections are reprints of the abstracts of the four publications this thesis builds upon.

1.1.1. Expected Information Maximization: Using the I-Projection for Mixture Density Estimation

Philipp Becker, Oleg Arenz, and Gerhard Neumann (2020), Chapter 3.

Modeling highly multimodal data is a challenging problem in machine learning. Most algorithms are based on maximizing the likelihood, which corresponds to the M(oment)-projection of the data distribution to the model distribution. The M-Projection forces the model to average over modes it cannot represent. In contrast, the I(nformation)-projection ignores such modes in the data and concentrates on the modes the model can represent. Such behavior is appealing whenever we deal with highly multimodal data where modeling single modes correctly is more important than covering all the modes. Despite this advantage, the I-Projection is rarely used in practice because it lacks algorithms that can efficiently optimize it based on data. In this work, we present a new algorithm called Expected Information Maximization (EIM) for computing the I-Projection solely based on samples for general latent variable models, where we focus on Gaussian mixtures models and Gaussian mixtures of experts. Our approach applies a variational upper bound to the I-Projection objective, which decomposes the original objective into single objectives for each mixture component and the coefficients, allowing efficient optimization. Similar to GANs, our approach employs discriminators but uses a more stable optimization procedure, using a tight upper bound. We show that our algorithm is much more effective in computing the I-Projection than recent GAN approaches and we illustrate the effectiveness of our approach for modeling multimodal behavior on two pedestrian and traffic prediction datasets.

1.1.2. On Uncertainty in Deep State Space Models for Model-Based Reinforcement Learning

Philipp Becker and Gerhard Neumann (2022), Chapter 5.

Improved State Space Models, such as Recurrent State Space Models (RSSMs), are a key factor behind recent advances in model-based Reinforcement Learning (RL). Yet, despite their empirical success, many of the underlying design choices are not well understood. We show that RSSMs use a suboptimal inference scheme and that models trained using this inference overestimate the aleatoric uncertainty of the ground truth system. We find this overestimation implicitly regularizes RSSMs and allows them to succeed in model-based RL. We postulate that this implicit regularization fulfills the same functionality as explicitly modeling epistemic uncertainty, which is crucial for many other model-based RL approaches. Yet, overestimating aleatoric uncertainty can also impair performance in cases where accurately estimating it matters, e.g., when we have to deal with occlusions, missing observations, or fusing sensor modalities at different frequencies. Moreover, the implicit regularization is a side-effect of the inference scheme and not the result of a rigorous, principled formulation, which renders analyzing or improving RSSMs difficult. Thus, we propose an alternative approach building on well-understood components for modeling aleatoric and epistemic uncertainty, dubbed Variational Recurrent Kalman Network (VRKN). This approach uses Kalman updates for exact smoothing inference in a latent space and Monte Carlo Dropout to model epistemic uncertainty. Due to the Kalman updates, the VRKN can naturally handle missing observations or sensor fusion problems with varying numbers of observations per time step. Our experiments show that using the VRKN instead of the RSSM improves performance in tasks where appropriately capturing aleatoric uncertainty is crucial while matching it in the deterministic standard benchmarks.

1.1.3. KalMamba: Towards Efficient Probabilistic State Space Models for RL under Uncertainty

Philipp Becker, Niklas Freymuth, and Gerhard Neumann (2024a), Chapter 6.

Probabilistic State Space Models (SSMs) are essential for Reinforcement Learning (RL) from high-dimensional, partial information as they provide concise representations for control. Yet, they lack the computational efficiency of their recent deterministic counterparts such as S4 or Mamba. We propose KalMamba, an efficient architecture to learn representations for RL that combines the strengths of probabilistic SSMs with the scalability of deterministic SSMs. KalMamba leverages Mamba to learn the dynamics parameters of a Linear Gaussian SSM in a latent space. Inference in this latent space amounts to standard Kalman filtering and smoothing. We realize these operations using parallel associative scanning, similar to Mamba, to obtain a principled, highly efficient, and scalable probabilistic SSM. Our experiments show that KalMamba competes with state-of-the-art SSM approaches in RL while significantly improving computational efficiency, especially on longer interaction sequences.

1.1.4. Combining Reconstruction and Contrastive Methods for Multimodal Representations in RL

Philipp Becker, Sebastian Mossburger, Fabian Otto, and Gerhard Neumann (2024b), Chapter 7.

Learning self-supervised representations using reconstruction or contrastive losses improves performance and sample complexity of image-based and multimodal Reinforcement Learning (RL). Here, different self-supervised loss functions have distinct advantages and limitations depending on the information density of the underlying sensor modality. Reconstruction provides strong learning signals but is susceptible to distractions and spurious information. While contrastive approaches can ignore those, they may fail to capture all relevant details and can lead to representation collapse. For multimodal RL, this suggests that different modalities should be treated differently based on the amount of distractions in the signal. We propose Contrastive Reconstructive Aggregated representation Learning (CoRAL), a unified framework enabling us to choose the most appropriate self-supervised loss for each sensor modality and allowing the representation to better focus on relevant aspects. We evaluate CoRAL's benefits on a wide range of tasks with images containing distractions or occlusions, a new locomotion suite, and a challenging manipulation suite with visually realistic distractions. Our results show that learning a multimodal representation by combining contrastive and reconstruction-based losses can significantly improve performance and solve tasks that are out of reach for more naive representation learning approaches and other recent baselines.

2. Foundations

In this section, we will establish the theoretical foundations for this work. First, we will review the techniques we use for structuring probabilistic models throughout this thesis (Section 2.1) and how to learn such models from data (Section 2.2). Next, we will discuss how to use probabilistic models for world modeling and how to use these world models to learn behaviors through Reinforcement Learning (Section 2.3). Finally, we will examine how to use probabilistic models for Imitation Learning by distribution matching (Section 2.4).

Throughout this thesis, we will denote vectors using lowercase bold letters (e.g., $\mathbf{x}$), matrices using uppercase bold letters (e.g., $\mathbf{X}$), and scalars, as well as general unspecified elements, using lowercase letters (e.g., x). Furthermore, we will use $\mathbf{x}_{\leq T}$ to denote a sequence of vectors $(\mathbf{x}_t)_{t=1\cdots T}$ as well as $\mathbf{x}_{\leq t}$ and $\mathbf{x}_{\geq t}$ to denote parts of a sequence up to t or from t onwards respectively.

Unless explicitly cited otherwise, the content of Section 2.1 and Section 2.2 are based on *Machine Learning: A Probabilistic Perspective* (Murphy, 2012).

2.1. Probabilistic Modeling and Inference

Probabilistic models are at the heart of modern Machine Learning, where most methods are formulated as fitting either a marginal distribution $p(x)$ or a conditional distribution $p(x|y)$ from data. For the embodied agents considered in this thesis, such probabilistic models are not just a helpful tool but necessary. To act efficiently and safely, these agents must be able to reason about the uncertainty and unpredictability of their environment and human behavior. Probabilistic models provide a principled mathematical framework for modeling such uncertainty and reasoning about it. In this section, we will review how to construct them by imposing meaningful structures and how to use them for inference.

2.1.1. Giving Structure to Probabilistic Models

For many interesting models, there is no single random variable x but multiple ones, $x_{1:N} = \{x_1, \cdots, x_N\}$. For an embodied agent, these could represent sequences of sensor readings from multiple sensors, predictions about the current state of the world, or the results of actions taken by the agent. In such cases, naively modeling the unstructured joint distribution $p(x_{1:N})$ is often neither simple nor particularly useful. Instead, we want to impose structure on the model, allowing us to construct simpler, more scalable, and more interpretable models. Two fundamental concepts for imposing such a structure, which we extensively use in this thesis, are latent variables and probabilistic graphical models.

2.1.1.1. Latent Variable Models

Perhaps counterintuitively, we can often simplify probabilistic models by introducing additional random variables. So-called latent or hidden variables are not part of the original data and are typically assumed to model common causes that gave rise to some observed data points.

Formally, there can be an arbitrary number M of latent variables that we denote as $z_{1:M} = \{z_1, \cdots, z_M\}$. The resulting latent variable model of the observables $x_{1:N}$ is then given by

$$p(x_{1:N}) = \int p(x_{1:N}, z_{1:M}) dz_{1:M}. \tag{2.1}$$

Think about examples from the medical domain to gain an intuition about the value of such latent variable models. Here, a doctor can usually only observe or measure the symptoms of a patient, corresponding to the observable hidden variables $x_{1:N}$. However, to find the best treatment, the doctor is ultimately interested in the underlying disease, which is not directly observable and thus corresponds to the latent variables $z_{1:M}$.

2.1.1.2. Conditional Independence and Probabilistic Graphical Models

The next step in probabilistic modeling is to impose structure on the joint distribution in Equation 2.1. A key concept for imposing structure is conditional independence, i.e., assuming that two or more random variables do not affect each other, given knowledge about some other variables. Formally, we say that two random variables x_1 and x_2 are conditionally independent given z if

$$p(x_1, x_2|z) = p(x_1|z)p(x_2|z)$$

and denote this as $x_1 \perp\!\!\!\perp x_2|z$. We can naturally extend this notion to sets of random variables. For example $x_{1:k} \perp\!\!\!\perp x_{k+1:N}|z_{1:M}$. for some $k \in \{1, N\}$, denotes that the first k random variables are conditionally independent of the last $N - k$ random variables given the latent variables $z_{1:M}$. Returning to our previous medical example, it would be reasonable to assume that symptoms are conditionally independent given knowledge of the underlying disease, which should provide sufficient information to explain each symptom.

We can visualize such independence assumptions using a directed acyclic graph, leading to Directed Graphical Models (DGMs), also known as Bayesian Networks (Pearl, 1988;

Koller and Friedman, 2009). Here, the nodes of the graph represent the random variable, while the edges model the dependency structure between them. More specifically, given the edges, the joint distribution factorizes as

$$p(x_{1:N}) = \prod_{i=1}^{N} p(x_i|pa_i)$$

where p_i are the parents of x_i in the graph. In terms of conditional independence, this means that each node is conditionally independent of all its descendants given its parents.

To reason about conditional independence between two sets of random variables in the graph, given a third set of random variables, we can use the so-called d-separation criterion. This criterion states that two sets of random variables X and Y are conditionally independent given a third set of random variables Z if all paths between X and Y are blocked by Z. Formally, a path is blocked by Z if one of the following conditions holds:

1. The path contains a chain $X \rightarrow Z \rightarrow Y$ or a fork $X \leftarrow Z \rightarrow Y$ and Z is not in the conditioning set.
2. The path contains a collider $X \rightarrow Z \leftarrow Y$, and neither Z nor any of its descendants are in the conditioning set.

2.1.1.3. Inference

Given a structured latent variable model, many tasks require us to infer a distribution over the latent variables given the observables $p(z_{1:M}|x_{1:N})$. We often refer to this distribution as a belief over the latent variables. Such a belief can serve as the basis for decision-making. Additionally, as we will see (Section 2.2.1) inferring beliefs is crucial for learning latent variable models from data in the first place.

The simplest form of inference is applying Bayes' rule

$$p(z_{1:M}|x_{1:N}) = \frac{p(x_{1:N}|z_{1:M})p(z_{1:M})}{p(x_{1:N})}.$$

However, this is only possible if the random variables are discrete, like those in our medical example, and for the most simple continuous-valued cases. For most cases in the context of embodied agents, computing $p(z_{1:M}|x_{1:N})$ becomes intractable, and therefore we need to resort to approximations. We will discuss such approximate inference techniques further in Section 2.2.

In our running example, inference corresponds to the doctor making a diagnosis based on the observed symptoms. They can then use their belief over the underlying disease as a basis for decision-making, e.g., choosing a treatment.

2.1.2. Mixture Models

Mixture models are among the simplest and most commonly used latent variable models. They assume a set of D simpler models, referred to as components, which are combined into a single complex model using a weighted sum. Formally, we can model the weights as a categorical distribution $p(z) = \mathrm{Cat}(\pi_1, \cdots, \pi_D)$, and the components as a conditional distribution $p(x|z)$. The categorical weight distribution selects the component based on the index z_i with corresponding probability $p(z_i)$. We can compute the full density of the mixture model by marginalization

$$p(x) = \sum_{i=1}^{D} p(x|z_i)p(z_i).$$

Given enough components, such mixture models can generally model arbitrarily complex distributions. Despite their richness, they offer interpretability, as we can often analyze the contribution of each component individually. Furthermore, they are highly scalable, as we can parallelize most computations over the individual components.

The most common choice of component, and also the most relevant for this thesis, is the multivariate Gaussian distribution, giving rise to Gaussian Mixture Models (GMMs). Here, the components are given by $p(\mathbf{x}|\mathbf{z}_i) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_i, \Sigma_i)$, where $\boldsymbol{\mu}_i$ and Σ_i are the mean and covariance of the i-th component. Together with the categorical weight distribution $p(z)$, this gives rise to the full model

$$p(\mathbf{x}) = \sum_{i=1}^{D} \pi_i \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_i, \Sigma_i).$$

So far, we have only looked at mixture models for marginal distributions $p(x)$. However, for many tasks, we are also interested in conditional distributions $p(x|y)$, which model the relation between inputs y and outputs x. While we can consider arbitrary distributions for the components, we will again focus on multivariate Gaussians, resulting in Gaussian Mixture of Experts (MoE) (Jacobs et al., 1991) models. Besides choosing the family of distributions for the components, for conditional models, we can also decide on how to model the relationship between inputs y and the parameters of the components. While more classical approaches would use a linear model for the mean and a constant covariance, modern methods often use neural networks to map from y to both.

As the first core contribution of this thesis, Chapter 3 will introduce a new approach to learning such mixture models from data.

2.1.3. State Space Models

State Space Models (SSMs) are the go-to choice for probabilistic modeling of time-series data. For such time series, the observable random variables $x_{1:N}$ are given by a series of

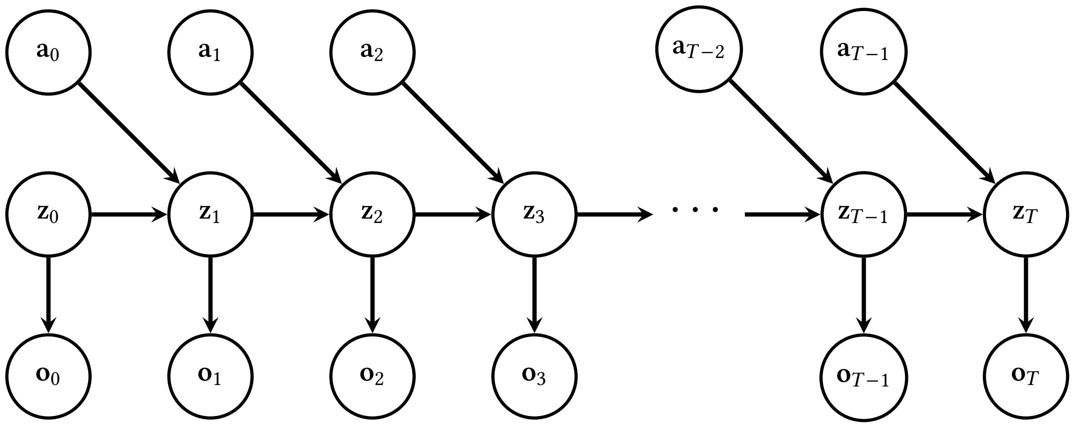


Figure 2.1.: Graphical model of a State Space Model (SSM). Each observation $\mathbf{o}_t$ depends only on the current latent state $\mathbf{z}_t$, and each latent state $\mathbf{z}_t$ depends only on the previous latent state $\mathbf{z}_{t-1}$ and the corresponding action $\mathbf{a}_{t-1}$.

observations $(\mathbf{o}_t)_{t=1\cdots T} = \mathbf{o}_{\leq T}$, with $\mathbf{o}_t \in \mathbb{R}^{d_o}$. SSMs assume this observation sequence is generated by a sequence of latent variables $(\mathbf{z}_t)_{t=1\cdots T} = \mathbf{z}_{\leq T}$, given inputs $(\mathbf{a}_t)_{t=1\cdots T} = \mathbf{a}_{\leq T}$. Here $\mathbf{z}_t \in \mathbb{R}^{d_z}$ and $\mathbf{a}_t \in \mathbb{R}^{d_a}$ are called the latent state and action at time t, respectively. For example, in robotics, the observations often correspond to sensor measurements of the robot and its surroundings, while the inputs are the control commands sent to the robot. To allow tractable modeling, SSMs make two conditional independence assumptions. First, each observation only depends on the current state. Second, each state only depends on the previous state and the corresponding action. Figure 2.1 shows the corresponding graphical model. Formally, the joint distribution of observations and latent states $p(\mathbf{z}_{\leq T}, \mathbf{o}_{\leq T} | \mathbf{a}_{\leq T})$ factorizes as

$$p(\mathbf{z}_{\leq T}, \mathbf{o}_{\leq T} | \mathbf{a}_{\leq T}) = p(\mathbf{z}_0) \left(\prod_{t=1}^{T} p(\mathbf{z}_t | \mathbf{z}_{t-1}, \mathbf{a}_{t-1}) \right) \left(\prod_{t=0}^{T} p(\mathbf{o}_t | \mathbf{z}_t) \right).$$

Here, the dynamics model $p(\mathbf{z}_t | \mathbf{z}_{t-1}, \mathbf{a}_{t-1})$ models how the latent states evolve over time, and the observation model $p(\mathbf{o}_t | \mathbf{z}_t)$ models how observations are generated from states. Finally, $p(\mathbf{z}_0)$ is the distribution over the initial state.

Given a SSM, there are several common inference problems. Filtering considers past observations $\mathbf{o}_{\leq t}$ and actions $\mathbf{a}_{\leq t}$ and aims to infer the current state $p(\mathbf{z}_t | \mathbf{o}_{\leq t}, \mathbf{a}_{\leq t})$. One example of such filtering is online tracking of a vehicle, where the current state is the vehicle's position and velocity, and the observations are sensor measurements such as GPS. If we do not need a fully online estimate, we can often improve the accuracy of our belief by also including future observations $\mathbf{o}_{\geq t}$ and actions $\mathbf{a}_{\geq t}$. Taking both past and future information into account to form a belief for time step t is referred to as smoothing and aims to infer the current state $p(\mathbf{z}_t | \mathbf{o}_{\leq T}, \mathbf{a}_{\leq T})$. Finally, given a belief about the current state, the SSM can predict distributions over future states and observations or draw samples from these distributions.

However, inference in SSMs is intractable for most models, and a vast body of literature exists to approximate inference techniques in SSMs (Julier and Uhlmann, 1997; Arulampalam et al., 2002). Next, we review the sole case where inference is tractable, namely Linear Gaussian SSMs. Section 2.2.2 discusses a common approximate inference technique for SSMs in a deep learning context, and Chapter 4, Chapter 5, and Chapter 6 introduce new methods for inference in SSMs, which combine classical techniques with modern deep learning approaches in a principled way as basis for world modeling and behavior learning.

2.1.3.1. Kalman Filtering and Smoothing: Inference in Linear Gaussian SSMs

The classical Kalman filtering (Kalman, 1960) and smoothing (Rauch et al., 1965) algorithms can viewed as inference in Linear Gaussian State Space Models (LGSSMs). Formally, LGSSMs assume dynamics and observation models which are (locally) linear and Gaussian, as well as an Gaussian initial state distribution,

$$p(\mathbf{z}_0) = \mathcal{N}(\mathbf{z}_0|\mathbf{0}, \Sigma^{\text{init}}) \tag{2.2}$$

$$p(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{a}_{t-1}) = \mathcal{N}(\mathbf{z}_t|\mathbf{A}_t\mathbf{z}_{t-1} + \mathbf{B}_t\mathbf{a}_{t-1}, \Sigma_t^{\text{dyn}}) \tag{2.3}$$

$$p(\mathbf{o}_t|\mathbf{z}_t) = \mathcal{N}(\mathbf{o}_t|\mathbf{H}_t\mathbf{z}_t, \Sigma_t^{\text{obs}}), \tag{2.4}$$

where $\mathbf{A}_t, \mathbf{B}_t, \mathbf{H}_t$ are matrices, $\Sigma_t^{\text{dyn}}, \Sigma_t^{\text{obs}}, \Sigma^{\text{init}}$ are covariance matrices, and $\mathbf{0}$ is a vector of zeros. These assumptions allow inference of filtered and smoothed beliefs in closed form.

The Kalman filter (Kalman, 1960) works by iteratively computing the posterior distribution over the latent state $p(\mathbf{z}_t|\mathbf{o}_{\leq t}, \mathbf{a}_{\leq t-1})$ given the current observation $\mathbf{o}_t$ and the previous state estimate $p(\mathbf{z}_{t-1}|\mathbf{o}_{\leq t-1}, \mathbf{a}_{\leq t-2})$,

$$p\left(\mathbf{z}_t|\mathbf{o}_{\leq t}, \mathbf{a}_{\leq t-1}\right) = \frac{1}{Z} p\left(\mathbf{o}_t|\mathbf{z}_t\right) p\left(\mathbf{z}_t|\mathbf{o}_{\leq t-1}, \mathbf{a}_{\leq t-1}\right) \tag{2.5}$$

$$= \underbrace{\frac{1}{Z} p\left(\mathbf{o}_t|\mathbf{z}_t\right)}_{\text{update}} \underbrace{\int p\left(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{a}_{t-1}\right) p\left(\mathbf{z}_{t-1}|\mathbf{o}_{\leq t-1}, \mathbf{a}_{\leq t-2}\right) d\mathbf{z}_{t-1}}_{\text{predict}} . \tag{2.6}$$

As indicated, this equation consists of two steps, prediction and update.

First, the predict step uses the previous belief $p(\mathbf{z}_{t-1}|\mathbf{o}_{\leq t-1}, \mathbf{a}_{\leq t-2}) = \mathcal{N}(\mathbf{z}_{t-1}|\boldsymbol{\mu}_{t-1}^{-}, \Sigma_{t-1}^{-})$, the dynamics model $p(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{a}_{t-1})$, and the action $\mathbf{a}_{t-1}$ to propagate the belief forward in time. This leads to a prior (to the observation) belief over the latent state $p(\mathbf{z}_t|\mathbf{o}_{\leq t-1}, \mathbf{a}_{\leq t-1}) = \mathcal{N}(\mathbf{z}_t|\boldsymbol{\mu}_t^{-}, \Sigma_t^{-})$ whose parameters are given by

$$\boldsymbol{\mu}_{t+1}^{-} = \mathbf{A}_t\boldsymbol{\mu}_t^{+} \quad \text{and} \quad \Sigma_{t+1}^{-} = \mathbf{A}_t\Sigma_t^{+}\mathbf{A}_t^T + \Sigma_t^{\text{dyn}}.$$

Next, the prior belief is updated using the current observation $\mathbf{o}_t$ to obtain the current filtered belief $p(\mathbf{z}_t|\mathbf{o}_{\leq t}, \mathbf{a}_{\leq t-1}) = \mathcal{N}(\mathbf{z}_t|\boldsymbol{\mu}_t^{+}, \Sigma_t^{+})$. Using Bayes's rule for Gaussian distributions, this leads to

$$\boldsymbol{\mu}_t^{+} = \boldsymbol{\mu}_t^{-} + \mathbf{Q}_t\left(\mathbf{o}_t - \mathbf{H}_t\boldsymbol{\mu}_t^{-}\right) \text{ and } \Sigma_t^{+} = \left(\mathbf{I} - \mathbf{Q}_t\mathbf{H}_t\right)\Sigma_t^{-} \text{ with } \mathbf{Q}_t = \Sigma_t^{-}\mathbf{H}^T\left(\mathbf{H}_t\Sigma_t^{-}\mathbf{H}_t^T + \Sigma_t^{\text{obs}}\right)^{-1},$$

where $\mathbf{I}$ denotes the identity matrix. The matrix $\mathbf{Q}_t$ is referred to as the Kalman gain can be interpreted as the weight for a weighted average between state and observation estimates. This weighting takes the uncertainty into account to update the belief based on the trust in both state and observation.

If we are interested in the smoothed belief, we can compute them by iterating backwards over the sequence and reusing quantities computed during filtering. While multiple versions of this so called backward pass exist, we will focus on the most basic and widely used one, resulting in the Rauch-Tung-Striebel (RTS) smoother (Rauch et al., 1965). The RTS smoother computes the smoothed belief $p(\mathbf{z}_t|\mathbf{o}_{\leq T}, \mathbf{a}_{\leq T})$ using the filtered belief $p(\mathbf{z}_t|\mathbf{o}_{\leq t}, \mathbf{a}_{\leq t-1})$ and the smoothed belief from the next time step $p(\mathbf{z}_{t+1}|\mathbf{o}_{\leq T}, \mathbf{a}_{\leq T})$ as

$$p\left(\mathbf{z}_t|\mathbf{o}_{\leq T}, \mathbf{a}_{\leq T}\right) = p(\mathbf{z}_t|\mathbf{o}_{\leq t}, \mathbf{a}_{\leq t-1}) \int \frac{p(\mathbf{z}_{t+1}|\mathbf{z}_t, \mathbf{a}_t) p\left(\mathbf{z}_{t+1}|\mathbf{o}_{\leq T}, \mathbf{a}_{\leq T}\right)}{p(\mathbf{z}_{t+1}|\mathbf{o}_{\leq t}, \mathbf{a}_{\leq t})} d\mathbf{z}_{t+1}. \tag{2.7}$$

We can naturally initialize this backward recursion, by realizing that the filtered and smoothed beliefs are equivalent at the last time step, as no future information is available. If we denote the parameters of the smoothed belief as $\boldsymbol{\mu}_t^s$ and Σ_t^s, this equation gives rise to the following recursion

$$\boldsymbol{\mu}_t^s = \boldsymbol{\mu}_t^+ + \mathrm{C}_t \left(\boldsymbol{\mu}_{t+1}^s - \boldsymbol{\mu}_{t+1}^-\right) \text{ and } \Sigma_t^s = \Sigma_t^+ + \mathrm{C}_t \left(\Sigma_{t+1}^s - \Sigma_{t+1}^-\right) \mathrm{C}_t^T \text{ with } \mathrm{C}_t = \Sigma_t^+ \mathrm{A}_t^T \left(\Sigma_{t+1}^-\right)^{-1}.$$

Here, the matrix C_t plays a similar role as the Kalman gain in the filtering step, as it trades off the contribution from past and future information.

2.1.3.2. Temporally Parallelized Kalman Filtering and Structured State Space Models

Forming belief states online is an inherently sequential process, which is reflected in the standard formulation of Kalman filtering as alternating update and predict steps. However, a key feature of modern deep learning models for time series, such as transformers (Vaswani et al., 2017), is their ability to process the entire sequence in parallel during training by parallelizing over the temporal dimension. Given parallel hardware, such as GPUs, temporal parallelism enables them to consume massive amounts of data and learn from long sequences.

One way to achieve such temporal parallelism is binary associative scanning, also known as parallel scanning or the prefix sum algorithm (Blelloch, 1990; Harris et al., 2007). For an arbitrary sequence of values $(x_t)_{t=1\cdots T}$ and an binary associative operation $\oplus$ this algorithm allows computing the sequence

$$(x_1, x_1 \oplus x_2, \cdots, x_1 \oplus x_2 \oplus \cdots \oplus x_T)$$

in parallel, i.e., in $O(\log T)$ time using $O(T)$ processors. Intuitively, parallel scanning exploits the associativity of $\oplus$ to rearrange the computations and compute the terms in parallel using two passes over the sequence. The first pass recursively combines adjacent pairs of elements, halving the number of elements in each step. Then, the second pass combines the partial results to construct the entire sequence. If $\oplus$ is a simple addition, this approach computes the cumulative sum of the elements.

In modern deep learning, such parallel scanning is one primary driver behind a recent class of methods known as Structured SSMs. Models such as S4 (Gu et al., 2021), S5 (Smith et al., 2022), and Mamba (Gu and Dao, 2023) formulate a sequence model using the linear recurrence

$$\mathbf{h}_t = \mathbf{A}_t \mathbf{h}_{t-1} + \mathbf{B}_t \mathbf{x}_t,$$

where $\mathbf{h}_t$ is the state and $\mathbf{x}$ the input at time t and $\mathbf{A}_t$, $\mathbf{B}_t$ are the learnable model parameters. Different models of this class differ in how exactly they parameterize $\mathbf{A}_t$ and $\mathbf{B}_t$, as well as how they efficiently realize the recurrence on parallel hardware. While earlier models used convolutions (Gu et al., 2021), more recent models use the parallel scanning technique described above (Smith et al., 2022; Gu and Dao, 2023). However, unlike the previously discussed SSMs, these structured SSMs are entirely deterministic and designed as a general approach for sequence-to-sequence modeling.

Concurrently, Särkkä and García-Fernández (2020) exploited the parallel scanning technique for inference in LGSSMs by showing how to rewrite the Kalman filtering and smoothing equations in terms of binary associative operations. However, they only consider classical state estimation problems, assuming all models are known.

In Chapter 6 we combine these structured SSMs with the framework from Särkkä and García-Fernández (2020) to build a new deep probabilistic SSM capable of processing sequences in $O(\log T)$ time.

2.2. Variational Inference and Learning in Latent Variable Models

In the previous section, we saw how to utilize latent variables and probabilistic graphical models to impose structure on complex distributions and how to infer exact beliefs over the latent variables using Bayes' rule. However, for many problems in the context of embodied agents, computing the belief over the latent variables $p(z|x)$ in closed form is intractable, and we need to resort to approximate inference methods.

While many methods for approximate inference exist, this thesis will focus on Variational Inference (VI) . Generally, VI frames inference as an optimization problem, where we want to find a parametric distribution $q(x)$ that is as close as possible to an intractable distribution $p(x)$. More formally, in VI, we aim to minimize a distance or divergence between the approximate and true distributions. While we can use arbitrary distances and divergences, we will focus on the most common choice, namely the (reverse) Kullback-Leibler divergence (KL-divergence) (Kullback and Leibler, 1951), which aims to find an approximate posterior by solving

$$\arg\min_{q(x)} \mathrm{KL}\left(q(x)||p(x)\right) \quad \text{with} \quad \mathrm{KL}\left(q(x)||p(x)\right) = \int q(x) \log \frac{q(x)}{p(x)} dx. \tag{2.8}$$

In a modern machine learning context, this essentially reframes the inference problem as a learning problem. This perspective on approximate inference as optimization enables us to apply the same techniques to two related challenges central to this thesis.

First, we can use VI to approximate target distributions for which we have access to an unnormalized density function $\hat{p}(x)$, i.e., a function that specifies the distribution $p(x)$ up to its normalization constant Z, as discussed in Section 2.2.4. This form of VI is a central prerequisite for the distribution matching approach for behavior learning by imitation we will introduce in Chapter 3.

For the second application of VI, we will focus on learning latent variable models from data. In this setting, we need VI to obtain a tractable training objective that circumvents marginalizing the latent variables. In Section 2.2.1 and Section 2.2.2, we will establish a framework for learning latent variable models from sequential data using VI. Additionally, in Section 2.2.3, we will examine an alternative yet related approach to learning latent variable models from data by maximizing mutual information. This framework will serve as the basis for the world models introduced in Chapter 5 and Chapter 6, as well as the training approach presented in Chapter 7.

2.2.1. Learning Latent Variable Models from Data with Variational Inference

Straightforward optimization of a latent variable model's log-likelihood

$$\log p(x) = \log \int p(x|z)p(z)dz$$

is usually not possible as computing the likelihood involves marginalization over the latent variables, which is generally intractable. An easier alternative would be to optimize the joint log-likelihood of the observable and latent variables, $\log p(x, z)$. However, it is not immediately clear that this leads an equivalent solution. Additionally, it requires knowledge about the latent variables, which leads to a "chicken-and-egg" problem, as inferring the latent variables depends on the model.

As a remedy, the classical Expectation-Maximization (EM) (Dempster et al., 1977) algorithm proposes a lower bound objective to show the equivalence and circumvent the "chicken-and-egg" problem by breaking it into two iterative steps. We can obtain this lower bound, usually called the Evidence Lower Bound (ELBO), by introducing a variational distribution $q(z)$ and rewriting the marginal log-likelihood of the observable variables as

$$\log p(x) = \underbrace{\mathbb{E}_{q(z)}\left[\log p(x|z)\right] - \mathrm{KL}\left(q(z)||p(z)\right)}_{\text{ELBO}} + \underbrace{\mathrm{KL}\left(q(z)||p(z|x)\right)}_{\geq 0}. \tag{2.9}$$

$$= \underbrace{\mathbb{E}_{q(z)}\left[\log p(x, z)\right] + H\left(q(z)\right)}_{\text{ELBO}} + \underbrace{\mathrm{KL}\left(q(z)||p(z|x)\right)}_{\geq 0}, \tag{2.10}$$

where $H(q(z)) = -\int q(z)\log q(z)dz$ is the entropy of the variational distribution $q(z)$. Here, we present two equivalent ways of writing the ELBO, which are useful in different contexts. In both formulations, we can see how the ELBO bounds the marginal log-likelihood of the observable variables x from below as the KL-divergence term in $\log p(x) = \text{ELBO} + \text{KL}(q(z)||p(z|x))$ is always non negative. Thus, any increase in the ELBO will also increase the marginal log-likelihood of the observable variables x, provided that the bound is tight, i.e., $q(z) = p(z|x)$.

The EM algorithm uses this relationship to optimize the ELBO iteratively. First, during the E-Step, the bound is tightened by keeping the model fixed and minimizing the KL-divergence term $\text{KL}(q(z)||p(z|x))$ for each x in the dataset. This step is an instance of the general VI problem in Equation 2.8, and its exact realization depends on the model and choice of the variational distribution $q(z)$. Next, during the M-Step, and fix the variational distribution $q(z)$. For such fixed variational distribution $q(z)$, optimizing the ELBO in Equation 2.10 is equivalent to maximizing the joint log-likelihood of the observable and latent variables, as the entropy term is constant with respect to $p(x)$. This resolves the "chicken-and-egg" problem, as we can now use the variational distribution $q(z)$ instead of the unknown latent variables z to optimize the model.

For example, in the simple case of GMMs, we can compute $p(z|x)$ in closed form using Bayes' rule during the E-Step. During the M-Step, we now update the weight distribution $p(z)$ using maximum likelihood, and the parameters of each Gaussian component are updated using a weighted version of maximum likelihood. Here the so called responsibilities $p(z|x)$, computed during the E-Step, serve as weights. If such closed-form solutions are not available, we need to resort to approximation-driven versions of the E-step (Neal and Hinton, 1998). More generally, many approaches exist that approximate either the M-Step with numerical optimization, the E-Step with VI, or both (McLachlan and Krishnan, 2008). However, here we will focus on the predominant approach in modern deep learning, namely stochastic gradient variational bayes (Kingma and Welling, 2013), which learns both the generative model and an approximate variational distribution jointly using stochastic gradient descent.

2.2.1.1. Stochastic Gradient Variational Bayes and Variational Autoencoding

The stochastic gradient variational Bayes approach (Kingma and Welling, 2013) combines the idea of optimization-based approximations to the EM algorithm with deep learning. Building on the ELBO in Equation 2.9, Kingma and Welling (2013) make two major changes over the classical EM-like approach. First, they use an amortized inference distribution $q(z|x)$, i.e., instead of approximating $p(z|x)$ for each sample x separately, they train a model mapping from x to z using a parametric family of distributions, e.g., a Gaussian distribution with mean and variance given by a neural network. Second, they discard the alternating nature of the EM approach and instead jointly optimize the parameters of the generative and amortized inference distribution using stochastic gradient descent on the

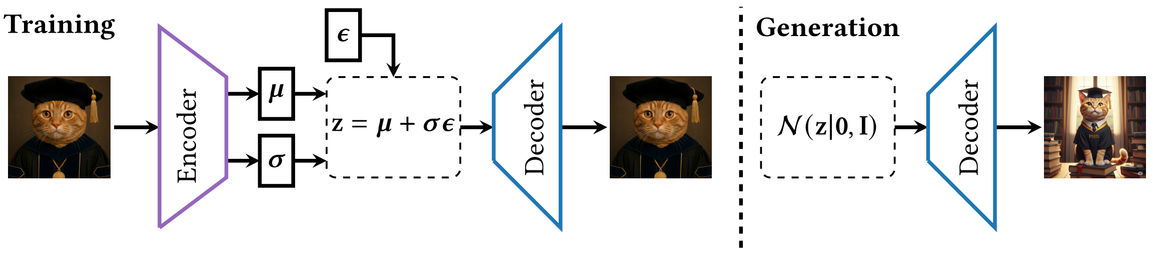

Training
Encoder
μ
σ
ϵ
$\mathbf{z} = \mu + \sigma\epsilon$
Decoder
Generation
$\mathcal{N}(\mathbf{z}|\mathbf{0}, \mathbf{I})$
Decoder

Figure 2.2.: During **training**, the VAE encodes an image $\mathbf{x}$ to extract the parameters of a Gaussian amortized inference distribution $q(z|\mathbf{x})$. Next, the parameters of this distribution are used to sample a latent variable $\mathbf{z}$ using the reparameterization trick. The latent variable $\mathbf{z}$ is then decoded to reconstruct the image using the generative model $p(\mathbf{x}|\mathbf{z})$. The parameters of the generative model are optimized to maximize the ELBO, which consists of two terms: the expected log-likelihood of the reconstruction and the KL-divergence between the amortized inference distribution and the prior distribution $p(\mathbf{z})$, which usually is an isotropic Gaussian with unit variance, $\mathcal{N}(\mathbf{z}|\mathbf{0}, \mathbf{I})$. After training, the VAE can **generate** new images by sampling from the prior distribution $p(\mathbf{z})$ and decoding the latent variable $\mathbf{z}$ using the generative model $p(\mathbf{x}|\mathbf{z})$.

ELBO. With the amortized inference distribution, the ELBO for a single sample x can be written as

$$\mathcal{L}_{\text{svgb}}(x) = \mathbb{E}_{q(z|x)}\left[\log p(x|z)\right] - \mathrm{KL}\left(q(z|x)||p(z)\right).$$

To allow optimizing this objective using stochastic gradient descent, Kingma and Welling (2013) introduce a stochastic gradient estimator based on the reparameterization trick (Kingma and Welling, 2013; Rezende et al., 2014). This trick reframes sampling from a distribution with given parameters as sampling from a fixed distribution and then deterministically transforming the sample using these parameters. For example, suppose z follows a Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$. In that case, we can sample from a standard normal distribution $\epsilon \sim \mathcal{N}(0, 1)$ and then transform this sample to obtain a sample from the desired distribution as $z = \mu + \sigma \cdot \epsilon$.

Kingma and Welling (2013) instantiate this approach for latent variable models by using deep neural networks to parameterize the generative model $p(x|z)$ and the amortized inference distribution $q(z|x)$, resulting in the Variational Autoencoder (VAE). Figure Figure 2.2 illustrates the VAE architecture for image data. While image compression and generation is still the most common use case for modern versions of VAEs (Van Den Oord et al., 2017; Rombach et al., 2022), the general framework has found applications in many other domains. In particular, when combined with State Space Models (SSMs), VAEs provide a powerful framework for learning probabilistic world models from sequential data.

2.2.2. Sequential Variational Autoencoding for World Modeling

To learn sequential world models from observations $\mathbf{o}_{\leq T}$, given actions $\mathbf{a}_{\leq T}$ we need to extend the VAE framework to the conditional setting (Sohn et al., 2015) as well as sequential data. For this, we first replace the latent variable z with a sequence of latent state vectors

$\mathbf{z}_{\leq T}$, one for each time step. Additionally, conditioning the entire model on the actions gives us the following lower-bound objective

$$\log p(\mathbf{o}_{\leq T}|\mathbf{a}_{\leq T}) = \underbrace{\mathrm{KL}\left(q(\mathbf{z}_{\leq T}|\mathbf{o}_{\leq T}, \mathbf{a}_{\leq T})||p(\mathbf{z}_{\leq T}|\mathbf{o}_{\leq T}, \mathbf{a}_{\leq T})\right)}_{\geq 0}$$

$$= \underbrace{\mathbb{E}_{q(\mathbf{z}_{\leq T}|\mathbf{o}_{\leq T},\mathbf{a}_{\leq T}))}\left[\log p(\mathbf{o}_{\leq T}|\mathbf{z}_{\leq T}|\mathbf{a}_{\leq T})\right] - \mathrm{KL}\left(q(\mathbf{z}_{\leq T}|\mathbf{o}_{\leq T}, \mathbf{a}_{\leq T})||p(\mathbf{z}_{\leq T}|\mathbf{a}_{\leq T})\right)}_{\text{Lower Bound}}$$

where $q(\mathbf{z}_{\leq T}|\mathbf{o}_{\leq T}, \mathbf{a}_{\leq T})$ is the amortized inference distribution. To get a model that is both more tractable and more useful, we can now introduce the State Space assumptions (Section 2.1.3) into this model.

This general approach yields a powerful framework for learning world models for behavior learning. The following section will introduce one specific approach, namely the Recurrent State Space Model (RSSM) (Hafner et al., 2019). Furthermore, we will look at how to learn behaviors on top of such models in Section 2.3. These foundations lay the groundwork for Chapter 5, Chapter 6, and Chapter 7 where we will introduce improved approaches to learn such world models using variational autoencoding as the basis for behavior learning.

2.2.2.1. Recurrent State Space Models

The Recurrent State Space Model (RSSM) (Hafner et al., 2019) is a specific instantiation of a sequential Variational Autoencoder for world modeling.

Its generative model generally follows the state space assumptions but splits the latent state into a deterministic $\mathbf{h}_t$ and stochastic part $\boldsymbol{\zeta}_t$. Given this split, Hafner et al. (2019) define the generative model as

$$h_t = f\left(h_{t-1}, \boldsymbol{\zeta}_{t-1}, \mathbf{a}_{t-1}\right) \quad \mathcal{N}\left(\boldsymbol{\zeta}_t|\boldsymbol{\mu}_t(\mathbf{h}_t), \boldsymbol{\sigma}(\mathbf{h}_t)\right)$$

where f is a recurrent neural network consisting of a Gated Recurrent Unit (GRU) (Cho et al., 2014) and $\boldsymbol{\mu}_t$ and $\boldsymbol{\sigma}_t$ are neural networks that output the mean and standard deviation of the Gaussian distribution for the stochastic part of the latent state. Additionally, they define the observation model as $p(\mathbf{o}_t|\mathbf{h}_t, \boldsymbol{\zeta}_t) = \mathcal{N}(\mathbf{o}_t|\boldsymbol{\mu}(\mathbf{h}_t, \boldsymbol{\zeta}_t), \sigma\mathbf{I}))$, where $\boldsymbol{\mu}(\mathbf{h}_t, \boldsymbol{\zeta}_t)$ is a neural network which is shared across time steps. According to Hafner et al. (2019), the split is necessary as a purely stochastic latent state is unable to reliably capture the model's dynamics and memorize information over long time horizons.

The RSSM's inference model shares large parts of the generative model but adds a variational distribution

$$q\left(\boldsymbol{\zeta}_t|\mathbf{h}_{\leq t}, \mathbf{o}_{\leq t}\right) = \mathcal{N}\left(\boldsymbol{\zeta}_t|\boldsymbol{\mu}_t(\mathbf{o}_t, \mathbf{h}_t), \boldsymbol{\sigma}(\mathbf{o}_t, \mathbf{h}_t)\right)$$

conditioned on the deterministic part of the latent state $\mathbf{h}_{\leq t}$ and the observations $\mathbf{o}_{\leq t}$. This way of formulating the inference model results in a filtering inference, assuming the variational distribution factorizes as $q(\boldsymbol{\zeta}_{\leq T}|\mathbf{o}_{\leq T}, \mathbf{a}_{\leq T}) = \prod_{t=1}^{T} q(\boldsymbol{\zeta}_t|\mathbf{h}_t, \mathbf{o}_t)$.

As we will discuss in Chapter 5, this inference assumption has significant implications for how the model represents uncertainty and learns.

2.2.3. Using Mutual Information to Learn Latent Variable Models

So far, we have focused on generative approaches to learning latent variable models using VI, where we aim to obtain latent variables that contain sufficient information to reconstruct the observations. Such reconstruction-based training usually includes objectives that maximize the likelihood of the observations given the latent variable, i.e., $\log p(z)$. While this provides a strong signal to learn a latent probabilistic representation, it forces this representation to capture all parts of the observation $\mathbf{o}$, even those that are pure noise or irrelevant to the task at hand. Here, contrastive learning approaches provide a powerful alternative. Those use a discriminative objective and, intuitively, instead of learning to reconstruct the entire observation, they learn to encode the information necessary to determine if a sample is different from another sample. This approach offers a powerful method for learning representations that are invariant to irrelevant or distracting features within the data.

In this thesis, we will focus on contrastive approaches based on maximizing the Mutual Information (MI) between the observation $\mathbf{o}$ and the latent variable $\mathbf{z}$, $I(\mathbf{o}, \mathbf{z})$. Intuitively, the mutual information quantifies how much information about the observation $\mathbf{o}$ is contained in the latent variable $\mathbf{z}$ by measuring the reduction in uncertainty about $\mathbf{o}$ when knowing $\mathbf{z}$. Formally, the mutual information is defined as

$$\begin{aligned} I\left(\mathbf{o}, \mathbf{z}\right) = & \int p(\mathbf{o}, \mathbf{z}) \log \frac{p(\mathbf{o}, \mathbf{z})}{p(\mathbf{o})p(\mathbf{z})} d\mathbf{o}d\mathbf{z} \\ = & H\left(p(\mathbf{o})\right) - \mathbb{E}_{p(\mathbf{z})}\left[H\left(p(\mathbf{o}|\mathbf{z})\right)\right] = H\left(p(\mathbf{z})\right) - \mathbb{E}_{p(\mathbf{o})}\left[H\left(p(\mathbf{z}|\mathbf{o})\right)\right]. \end{aligned} \tag{2.11}$$

Formulating the MI as the difference between the entropy of the observation $\mathbf{o}$ and the conditional entropy of the observation given the latent variable $\mathbf{z}$, $H\left(\mathbf{o}\right) - H\left(\mathbf{o}|\mathbf{z}\right)$ formalizes the intuition that MI quantifies the reduction in uncertainty about the observation $\mathbf{o}$ when knowing the latent variable $\mathbf{z}$. Additionally, it immediately shows that MI is symmetric, i.e., $I\left(\mathbf{o}, \mathbf{z}\right) = I\left(\mathbf{z}, \mathbf{o}\right)$. The information an observation contains about its latent representation is identical to the information the latent representation contains about the observation.

As we will see in Chapter 7, the previously discussed reconstruction-based approaches and the MI-based approaches discussed here can be interchanged, and combining them in a principled way allows us to learn representations that enable learning behaviors in scenarios with challenging multi-sensor, observations. While Chapter 7 takes a pragmatic but still principled approach to combine MI and reconstruction-based objectives for learning latent variable models, there is a deeper connection between MI and the previously discussed variational autoencoding approaches through the information bottleneck (Tishby et al., 1999) objective and its deep learning instantiation, the Deep Variational Information Bottleneck (Alemi et al., 2016).

However, to use the MI in the first place, we must be able to compute it, which is not a straightforward task.

2.2.3.1. Estimating Mutual Information for Sequential Data

While maximizing MI is a powerful objective, directly computing it using Equation 2.11 would require learning a full generative model of the joint distribution $p(\mathbf{o}, \mathbf{z})$, which is precisely what contrastive methods seek to avoid. Fortunately, there exist multiple more effective approximations. Most notably, the InfoNCE (Noise Contrastive Estimation) (Oord et al., 2018) estimator provides a practical and alternative lower bound to MI, which we can optimize instead.

The core idea behind InfoNCE is to train a classifier that can distinguish one positive observation $\mathbf{o}^{\text{pos}}$ from a set of $N-1$ negative samples $\{\mathbf{o}_i^{\text{neg}}\}_{i=1:N-1}$ given the corresponding latent $\mathbf{z}$. This classifier is parameterized using a score function $f(\mathbf{o}, \mathbf{z})$, which should assign high values to positive pairs and low values to negative pairs. The resulting lower bound can then be written as

$$I(\mathbf{o}, \mathbf{z}) \geq \mathbb{E}_{p\left(\mathbf{o}^{\text{pos}}, \{\mathbf{o}_i^{\text{neg}}\}_{i=1:N-1}, \mathbf{z}\right)} \left[\log \frac{\exp f(\mathbf{o}^{\text{pos}}, \mathbf{z})}{\exp f(\mathbf{o}^{\text{pos}}, \mathbf{z}) + \sum_{i=1}^{N-1} \exp f(\mathbf{o}_i^{\text{neg}}, \mathbf{z})} \right].$$

For this approach to work, both designing a good score function $f(\mathbf{o}, \mathbf{z})$ and sampling distribution $p\left(\mathbf{o}^{\text{pos}}, \{\mathbf{o}_i^{\text{neg}}\}_{i=1:N-1}, \mathbf{z}\right)$ are crucial. The score function $f(\mathbf{o}, \mathbf{z})$ is typically implemented as a neural network that projects the observation and latent state into a shared embedding space before computing their similarity, for instance, with a dot product. To obtain the set of negative samples required by the objective, a common and highly efficient strategy is to use in-batch negatives. For a given positive pair $(\mathbf{o}_i, \mathbf{z}_i)$ all other observations $\mathbf{o}_j$ with $j \neq i$ within the same training mini-batch are treated as negative samples. This approach scales well with large batch sizes, providing a large and diverse set of negatives at little additional computational cost.

When using the InfoNCE estimator in the context of State Space Models (SSMs), the positive pairs are given by an observation $\mathbf{o}_t$ and the inferred latent state $\mathbf{z}_t$ at the same time step t, given the entire trajectory. In this case, we can extend the idea of in-batch negatives to in-sequence negatives, where the positive pair is given by the observation $\mathbf{o}_t$ and the latent state $\mathbf{z}_t$ at the same time step t. In contrast, the negative samples are given by all other observations $\mathbf{o}_k$ and latent states $\mathbf{z}_k$ at different time steps $k \neq t$ within the same sequence. If we use a batch of sequences, we can increase the number of negative samples by also including all observation-state pairs from the other sequences in the batch.

2.2.4. Variational Inference for Inference Given Unnormalized Densities

The second common use case for VI is learning a tractable distribution for inference and sampling given an unnormalized density function $\hat{p}(x)$. Such a function specifies a distribution $p(x)$ up to its normalization constant Z, i.e,

$$p(x) = \frac{\hat{p}(x)}{Z} \quad \text{with} \quad Z = \int \hat{p}(x) dx. \tag{2.12}$$

For many relevant problems, computing $\hat{p}(x)$ is much simpler than computing $p(x)$, as normalization requires solving the integral in Equation 2.12, which is often intractable. In this setting, VI can be used to fit a parametric distribution $q(x)$ given $\hat{p}(x)$ to obtain a tractable approximation of the target distribution $p(x)$ which can be used for sampling and inference. However, while tractable, evaluating the unnormalized density function $\hat{p}(x)$ can be costly, requiring VI approaches that emphasize sample efficiency.

The key insight towards finding a variational approximation lies in the fact that the KL-divergence between the variational distribution $q(x)$ and the unknown target density $p(x)$ can be rewritten as

$$\begin{aligned} \mathrm{KL}\left(q(x)||p(x)\right) &= \int q(x) \log \frac{q(x)}{\hat{p}(x)} dx - \log Z \\ &= -\int q(x) \log \hat{p}(x) dx - H(q(x)) - \log Z, \end{aligned} \tag{2.13}$$

where $H(q(x)) = -\int q(x) \log q(x) dx$ is the entropy of the distribution $q(x)$. Crucially, we can ignore the $\log Z$ term, as it is constant for the optimization, which results in an objective that only depends on $\hat{p}(x)$ and $q(x)$. While we can access the former by assumption, we are free to choose the model $q(x)$, allowing us to optimize the KL divergence with respect to $q(x)$.

This insight leaves us with two related design choices: which parametric family of distributions $q(x)$ to use and how to solve the optimization problem efficiently.

2.2.4.1. Variational Inference by Policy Search

For Variational Inference by Policy Search (VIPS), Arenz et al. (2018) choose Multivariate Gaussian Mixture Models (GMMs) as the parametric family

$$q(x) = \sum_{i=1}^{D} q(x|z_i) q(z_i).$$

However, while powerful, optimizing the resulting objective for such latent variable models is non-trivial. First, the entropy term $H\left(q(x)\right)$ is not tractable for mixture models. Second, optimizing the multivariate component distributions is hard, primarily due to the quadratic size and positive-definiteness constraints of the covariance matrices.

To address these optimization challenges, Arenz et al. (2018) propose an upper bound that yields individual updates for each mixture component and its corresponding weights.

For this upper bound, Arenz et al. (2018) first introduce an auxiliary distribution $q(\tilde{z}|x)$ allowing them to bound Equation 2.13 as

$$\begin{aligned} -\int q(x)\log\hat{p}(x)dx + H(q(x)) = &-\underbrace{\mathbb{E}_{q(x)}\left[\mathrm{KL}\left(q(z|x)||\tilde{q}(z|x)\right)\right]}_{\geq 0} \\ &\underbrace{-\int q(x)(\log p(x) + \log\tilde{q}(z|x))dxdz - \mathbb{E}_{q(z)}\left[H\left(q(x|z)\right)\right] - H\left(q(z)\right)}_{\text{upper bound}}, \end{aligned} \tag{2.14}$$

where the second part is an upper bound to the original objective as the KL term is non-negative. This bound holds for every choice of $\tilde{q}(z|x)$. However, the bound is only tight if $\tilde{q}(z|x) = q(z|x)$, which suggests an iterative approach, similar to the Expectation-Maximization (EM) algorithm (Dempster et al., 1977) (Section 2.2.1). For VIPS, in what corresponds to the E-step, we update the auxiliary distribution $\tilde{q}(z|x)$ using the current GMM and Bayes' rule and, in the corresponding M-step, we update the GMM parameters by minimizing the upper bound.

To update the GMM parameters, Arenz et al. (2018) repurpose an existing, highly sample-efficient policy search algorithm called Model-Based Relative Entropy Stochastic Search (MORE) (Abdolmaleki et al., 2015). For this approach, they first fit a quadratic surrogate function to the samples drawn from the auxiliary distribution $\tilde{q}(z|x)$ and re-weighted using the unnormalized density $\hat{p}(x)$. They then use this surrogate function to update the components' parameters in closed form while also adhering to an information-theoretic constraint, ensuring stable updates. Appendix A.1.2 provides additional details on how the MORE algorithm is used in VIPS.

In their original paper, Arenz et al. (2018) also introduce techniques to adapt the number of mixture components and reuse samples to improve sample efficiency further. Later, they further refined their approach to optimize sample efficiency even further (Arenz et al., 2020) and extended their approach to a first-order setting, assuming access to the gradient of $\hat{p}(x)$ (Arenz et al., 2023). In Chapter 3 of this thesis, we extend VIPS to the density estimation case, i.e., we will not assume access to the unnormalized density function $\hat{p}(x)$ of a distribution $p(x)$, but only to samples from $p(x)$. Furthermore, we go beyond the marginal case and consider fitting conditional latent variable models.

2.3. Reinforcement Learning for Behavior Learning with Probabilistic World Models

At its core, Reinforcement Learning (RL) is about how intelligent agents can learn to make optimal decisions through interaction with their environment. Inspired by the way how humans and animals learn from experience, RL frames behavior learning as a trial-and-error process, where the agent receives a reward signal indicating the quality of its decisions.

This section addresses how to use the previously discussed probabilistic models to learn goal-directed behavior through Reinforcement Learning. First, Section 2.3.1, Section 2.3.2, and Section 2.3.3 will introduce the foundations of RL. Here, we will focus on the partially observable setting, which is crucial for embodied agents in realistic environments. We will also highlight the differences between model-free and model-based RL, as well as the role of uncertainty for the latter. Finally, Section 2.3.4 will introduce the concept of probabilistic world models and discuss how we can use them to learn behavior through RL. There, we will tie together the previously discussed modeling and learning approaches with the RL foundations into a framework for learning behavior under partial observability. This framework will be the basis for the three different approaches to learning behavior with probabilistic world models presented in Chapter 5, Chapter 6, and Chapter 7.

2.3.1. Foundations of Reinforcement Learning

This section will briefly introduce the foundations of RL in the fully observable case and serve as a primer for the subsequent sections on RL under partial observability and model-based RL.

Unless otherwise cited, this section is based on *Reinforcement Learning: An Introduction* (Sutton and Barto, 2018).

2.3.1.1. Learning Optimal Behavior in Markov Decision Processes

Formally, the RL problem is defined using a so-called Markov Decision Process (MDP) (Bellman, 1957). The literature presents several slightly different definitions of MDPs, depending on the problem at hand. However, here we will focus on MDPs with probabilistic dynamics given by a set of states $s \in \mathcal{S}$ and actions $a \in \mathcal{A}$, a transition distribution $p(s_{t+1}|s_t, a_t)$, a reward function $R(s_t, a_t)$, and an initial state distribution $p_0(s_0)$. While both states and actions can be discrete or continuous in principle, we will focus on continuous settings, usually assuming they are a subset of $\mathbb{R}^d$. Crucially, to get a tractable decision process, both the transition distribution and reward function impose structure on the MDP by assuming that the next state s_{t+1} and reward $R(s_t, a_t)$ depend only on the current state s_t and action a_t.

Given an MDP, the goal of the agent is to learn a distribution $\pi(a|s)$ that maximizes the expected cumulative reward, also known as return, over time

$$J(\pi) = \mathbb{E}_{p_0(s_0) \prod_{t=0}^{\infty} \pi(a_t|s_t) p(s_{t+1}|s_t, a_t)} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \right], \tag{2.15}$$

where $\gamma \in [0, 1)$ is a discount factor that trades off between optimizing long and short-term rewards. The distribution $\pi(a|s)$ is usually referred to as policy and describes the agent's behavior given a certain state.

2.3.1.2. Value Functions and Q-Functions

To effectively evaluate and improve policies, most RL algorithms rely on auxiliary functions that allow them to assess the quality of being in a certain state or taking a certain action. The most common of these functions are the value function $V(s_t)$ and the state-action-value function $Q(s_t, a_t)$, also known as the Q-function. The value function $V(s_t)$ describes the expected return when starting in state s_t and following a certain policy π thereafter

$$V_\pi(s_t) = \mathbb{E}_{\prod_{k=t}^{\infty} \pi(a_k|s_k)p(s_{k+1}|s_k,a_k)} \left[\sum_{k=t}^{\infty} \gamma^k R(s_k, a_k) \right]. \tag{2.16}$$

Analogously, the Q-function $Q(s_t, a_t)$ describes the expected return when starting in state s_t, taking action a_t, and then following a certain policy π thereafter

$$Q_\pi(s_t, a_t) = \mathbb{E}_{\prod_{k=t}^{\infty} \pi(a_{k+1}|s_{k+1})p(s_{k+1}|s_k,a_k)} \left[R(s_t, a_t) + \sum_{k=t+1}^{\infty} \gamma^k R(s_k, a_k) \right]. \tag{2.17}$$

The true benefits of these auxiliary functions only become apparent when realizing that we can rewrite them in a recursive manner as

$$\begin{aligned} V_\pi(s_t) &= \mathbb{E}_{\pi(a_t|s_t)} \left[R(s_t, a_t) + \gamma \mathbb{E}_{p(s_{t+1}|s_t,a_t)} \left[V_\pi(s_{t+1}) \right] \right] \quad \text{and} \\ Q_\pi(s_t, a_t) &= R(s_t, a_t) + \gamma \mathbb{E}_{\pi(a_{t+1}|s_{t+1})p(s_{t+1}|s_t,a_t)} \left[Q_\pi(s_{t+1}, a_{t+1}) \right], \end{aligned}$$

respectively. These recursive equations are known as the Bellman equations. They are the basis for many RL algorithms, as they significantly simplify the expectations in Equation 2.16 and Equation 2.17. Additionally, by realizing that any optimal policy π^* will always pick the actions that maximize the value or Q function, we can rewrite the Bellman equations to directly compute the value and Q functions for this optimal policy

$$\begin{aligned} V_{\pi^*}(s_t) &= \max_{a_t} \left[R(s_t, a_t) + \gamma \mathbb{E}_{p(s_{t+1}|s_t,a_t)} \left[V_\pi(s_{t+1}) \right] \right] \quad \text{and} \\ Q_{\pi^*}(s_t, a_t) &= R(s_t, a_t) + \gamma \mathbb{E}_{p(s_{t+1}|s_t,a_t)} \left[\max_{a_{t+1}} Q_\pi(s_{t+1}, a_{t+1}) \right]. \end{aligned} \tag{2.18}$$

These equations are known as the Bellman Optimality Equations. For MDPs with discrete states and actions, they allow us to compute the optimal policy using dynamic programming. However, to model the kinds of embodied agents we focus on, we need approaches able to work with arbitrary state and action spaces.

2.3.1.3. Deep Reinforcement Learning with Continuous State and Action Spaces

For continuous state and action spaces, we cannot use the dynamic programming approach to compute the optimal value functions and Q-functions. Instead, we need to rely on function approximation. Nevertheless, in this case, we can use the Bellman optimality equations to get a training signal for these function approximators. As the resulting approaches are very similar for value functions and Q-functions, we will focus on the latter in this section.

We know that the Q-function of the optimal policy π^* has to satisfy Equation 2.18, which, in principle, gives us a supervised learning problem. We can optimize any function approximator $Q_\pi(s_t, a_t)$ to minimize the error between the left and right-hand side of the Bellman optimality equation, known as the Bellman error. If we choose the squared error as our loss function, this results in the Mean Squared Bellman Error (MSBE) $L_{\text{MSBE}}(Q) =$

$$\mathbb{E}_{p_0(s_0)\prod_{t=0}^{\infty}\pi(a_t|s_t)p(s_{t+1}|s_t,a_t)}\Big[\big(\underbrace{R(s_t,a_t)+\gamma\mathbb{E}_{p(s'_{t+1}|s_t,a_t)}\big[\max_{a'_{t+1}} Q_\pi(s'_{t+1},a'_{t+1})\big]}_{\text{Bellman Target}}-Q_\pi(s_t,a_t)\big)^2\Big].$$

However, even with discrete actions and continuous states, directly optimizing the MSBE is infeasible, especially with large and powerful function approximators such as large neural networks. To use neural networks as function approximators, deep Q-learning approaches employ a range of techniques to make this feasible and efficient. Most notable are experience replay, which stores past transitions in a buffer and samples them randomly to break temporal correlations, and target networks that slowly update a copy of the Q-network to compute the Bellman Target, providing a stable reference point for learning (Mnih et al., 2013). Yet, state-of-the-art approaches go far beyond this and employ a range of additional techniques to mitigate systematic biases, stabilize training, and improve performance (Hessel et al., 2018; Badia et al., 2020).

If we consider the setting we focus on, continuous state and action spaces, another problem with the above approach arises, as we cannot simply take the maximum of all actions in the domain of the Q-function. This issue gives rise to the paradigm of actor-critic methods, which simultaneously learn a Q-function and a policy. Here, the policy is updated to maximize the Q-function and then used to compute the Bellman target for the Q-function.

2.3.1.4. Soft Actor-Critic

Out of the family of actor-critic methods, the Soft Actor-Critic (SAC) (Haarnoja et al., 2018) is the one most prominently used in this thesis (Chapter 6 and Chapter 7). SAC augments the standard RL objective in Equation 2.15 with a maximum entropy term

$$J(\pi) = \mathbb{E}_{p_0(s_0)\prod_{t=0}^{\infty}(\pi(a_t|s_t)p(s_{t+1}|s_t,a_t))}\left[\sum_{t=0}^{\infty}\gamma^t R(s_t,a_t) + \alpha H\left(\pi(\cdot|s_t)\right)\right]. \tag{2.19}$$

Here, α is a hyperparameter determining the importance of the entropy term. Maximizing entropy, along with the cumulative expected reward, can improve both the policy's exploration behavior and its robustness.

To derive a practical algorithm from this objective, SAC learns a soft Q-function that considers both the expected future rewards and the policy's future entropy, thereby balancing reward-seeking with exploration behavior.

When updating the policy, SAC exploits an interesting relation (Levine, 2018) between maximum entropy RL and the Variational Inference (VI) framework introduced in Section 2.2. Given a (soft) Q-function $Q(s_t, a_t)$, the optimal policy is given by the unnormalized density of a Boltzmann distribution

$$\pi^*(a_t|s_t) \propto \exp\left(\frac{1}{\alpha}Q(s_t, a_t)\right).$$

Given this target, Haarnoja et al. (2018) use VI to fit a parametric policy $\pi_\theta(a_t|s_t)$ to this unnormalized density using

$$\begin{aligned}\pi &= \arg\min_{\pi_\theta} \mathbb{E}_{p_b(s_t)} \mathrm{KL}\left(\pi(a_t|s_t) || \frac{1}{Z}\exp\left(\frac{1}{\alpha}Q(s_t, a_t)\right)\right) \\ &= \arg\max_{\pi_\theta} \mathbb{E}_{p_b(s_t)}\left[\pi_\theta(a_t|s_t)Q(s_t, a_t) + \alpha H\left(\pi(a_t|s_t)\right)\right].\end{aligned}$$

This approach shows an interesting connection to VIPS (Arenz et al., 2018) (Section 2.2.4.1) and the approach introduced in Chapter 3, which use the same relationship to repurpose the MORE (Abdolmaleki et al., 2015) algorithm, initially designed for maximum entropy policy search, to update their distributions, in a VI and distribution matching setting respectively.

2.3.2. Reinforcement Learning Under Partial Observability

The MDP framework described previously assumes that the agent has full access to the state of the environment. However, for realistic embodied agents, this is usually not the case.

2.3.2.1. Partially Observable Markov Decision Processes (POMDPs)

The Partially Observable Markov Decision Process (POMDP) (Astrom, 1965) is a generalization of the MDP framework that allows to model partially observable states. Similar to the MDP, a POMDP has a set of states $z \in \mathcal{Z}$ and actions $a \in \mathcal{A}$, a transition distribution $p(z_{t+1}|z_t, a_t)$, a reward function $R(z_t, a_t)$, and an initial state distribution $p_0(z_0)$. However, in a POMDP, the state is no longer directly observable but hidden. To indicate that we are now dealing with a latent variable, we changed the notation from s to z. Instead, the agent receives observations $o_t \in \mathcal{O}$, which are generated from the state via an observation distribution $p(o_t|z_t)$. Besides being noisy, this observation distribution can also lose arbitrary information about the state.

Crucially, as the agent does not have access to the state, it cannot be used as input to the policy anymore. Furthermore, simply using the observation o_t as input to the policy is also clearly insufficient for most cases of partial observability. Thus, to learn good behavior in a POMDP, we need to use a policy $\pi(a_t|o_{\leq t}, a_{\leq t-1})$ that takes the history of observations and actions into account.

2.3.2.2. Using State-Space Models to solve POMDPs

Conditioning the policy explicitly on the history of observations and actions is generally impractical, as the history grows with each time step, making it hard to learn or even represent the policy. The standard solution to this problem is to compress the history of observations and actions into a compact representation that is still sufficient for optimal decision-making (Sondik, 1971; Kaelbling et al., 1998).

The first step towards such representations is realizing that the POMDP formulation is closely related to the State Space Model (SSM) framework introduced in Section 2.1.3, as both make the same assumptions about the dynamics and observation distributions. Hence, a sufficient statistic for the history of observations and actions is given by the filtered belief $b(z_t) = p(z_t|o_{\leq t}, a_{\leq t-1})$. In the POMDP literature, $b(z_t)$ is commonly referred to as the belief state.

By working with a policy conditioned on this belief, $\pi(a_t|b(z_t))$, we can effectively reduce the history of observations and actions, obtaining a more tractable decision process. Additionally, we can reuse the Bayesian filtering techniques to compute this belief state given the history of observations and actions or recursively update it as new observations and actions are made.

Two special cases of this approach are fully discrete POMDPs and what is known as Linear Quadratic Gaussian (LQG) control. If the latent states, actions, and observations are all discrete, forming the required belief states is straightforward using the forward algorithm for hidden Markov models (Murphy, 2012). However, even for a finite number of discrete states, there are an infinite number of possible beliefs, which makes learning a policy significantly more complex. Formally, the belief state for fully discrete POMDPs is a categorical distribution whose parameters are continuous, and therefore, tabular RL approaches are no longer applicable (Kaelbling et al., 1998). For POMDPs with continuous spaces, closed-form inference of the belief state is only possible under the typical linear Gaussian assumptions underpinning the Kalman filter. Interestingly, if we additionally assume quadratic cost (i.e., negative rewards), it can be shown that feeding the mean of the belief state into a linear quadratic regulator (LQR) results in an optimal policy. This is known as the separation principle and results in an approach called Linear Quadratic Gaussian (LQG) control (Anderson and Moore, 2007).

However, these approaches not only make strong assumptions about the structure of the POMDP but also rely on exact knowledge of the transition and observation distributions for inference of the belief state. One major contribution of this thesis is the formulation of new learning approaches for Deep SSMs, which enable the effective and accurate computation of approximate belief states in more general settings encountered by embodied agents.

2.3.3. Model-Based Reinforcement Learning

The Reinforcement Learning (RL) approaches discussed so far all operate in a model-free setting, meaning they learn policies directly from interaction with the environment without explicitly modeling it. In contrast, model-based RL (MBRL) (Moerland et al., 2023) approaches aim to learn a model of the environment's dynamics first and then use this model to learn optimal behavior. Model-based RL promises that this approach is significantly more sample efficient as it allows conducting the expensive trial-and-error process of RL in a simulated environment. Additionally, learning environment models allows a natural transfer of experiences from one task to another (Hansen et al., 2024), as many aspects of the environment, such as fundamental physics or object permanence, are consistent across tasks.

Many MBRL approaches are rooted in more classical ideas from model-based control. Assuming accurate knowledge of the environment's dynamics, model-based control approaches such as Model Predictive Control (MPC) or the Linear Quadratic Regulator (LQR) can achieve exceptional performance and are widely adopted in robotics and control engineering (Mayne, 2014). However, in classic model-based control, this accurate system knowledge is typically provided by manual modeling from human engineers and system identification. For more complex systems, manually designing models of sufficient quality is often infeasible.

Here, the MBRL approach of learning the model on the go can serve as an alternative to overcome the limitations of manual modeling. However, MBRL is fundamentally more difficult than learning a model by supervised learning and then adding a model-based control approach on top. The challenges revolve around the fact that the performance of the current policy highly depends on the quality of the current model, which in turn depends on the quality of the data collected by the policy (Deisenroth et al., 2013; Moerland et al., 2023). The key to mitigating these issues is to use models that are aware of their uncertainties, i.e., they must be aware of what they do not know (Deisenroth and Rasmussen, 2011; Chua et al., 2018; Kidambi et al., 2020). Such awareness allows the model to mitigate the effects of the issues outlined below, namely unreliable predictions, a mismatch between the model and the policy, and a distribution shift.

First, even minor inaccuracies in the model's prediction of the next step can accumulate over time (Talvitie, 2014). Such compounding errors can cause simulated rollouts to diverge significantly from the dynamics of the real environment. This effect renders long-term planning and control infeasible, often forcing model-based RL approaches to rely on short-term planning horizons (Janner et al., 2019)

Second, another critical challenge is the inherent objective mismatch (Lambert et al., 2020). The dynamics model optimizes the next state prediction loss using a supervised learning objective, while the policy aims to maximize the expected cumulative reward. These two objectives are not necessarily aligned as a model that makes globally accurate predictions might lack the resolution needed for optimal control in critical regions of the state space. Similarly, a model that accurately predicts the next state locally might make arbitrary errors in parts of the state space it has not yet seen or for which there is very little data (Kurutach et al., 2018). This mismatch can lead to policies that exploit the model's flaws, causing them to perform well in simulation but fail catastrophically in the real environment.

Third, learning a model for MBRL suffers from a fundamental distribution shift, breaking many of the assumptions underlying standard supervised learning. As the policy improves, it naturally pushes the agent into new regions of the state space, constantly changing the data distribution and expanding its support. This exploratory data collection creates a unique trilemma when designing the model and its learning approach. First, the model must be accurate enough from the start to bootstrap a useful policy. Second, it must be flexible enough to adapt quickly to new data, keeping pace with the evolving policy. Third, it must be expressive enough to eventually capture the dynamics of the entire relevant state space required for an optimal policy. This trilemma demands a model that is robust against overfitting to sparse data early in training yet has sufficient capacity to represent a complex and expanding world as learning progresses.

These challenges make it clear that the first step towards awareness of uncertainty is to use a principled probabilistic world model and not rely on a single deterministic prediction. Moreover, particularly in the partially observable setting, we need a model that can disentangle different sources of uncertainty, such as uncertainty due to lack of training data and uncertainty caused by the inherent randomness of the POMDP. This distinction is crucial when learning behavior. It allows the agent to adapt its behavior to its uncertainty about the environment while interacting with it without overly relying on the model during training if this is not warranted. Designing such probabilistic world models building on the foundations of SSMs is the key focus of Chapter 5 and Chapter 6.

2.3.3.1. Uncertainty in Model-Based Reinforcement Learning

Uncertainty in model-based RL (MBRL), and machine learning more generally, can be broadly categorized into epistemic and aleatoric uncertainty (Hüllermeier and Waegeman, 2021).

Epistemic uncertainty captures the model uncertainty about the world due to a lack of training data. In the context of MBRL, epistemic uncertainty arises from the model's lack of knowledge about the environment, particularly in regions of the state-action space that it has not explored sufficiently. Crucially, this uncertainty can be reduced by collecting more training data, and given a ground truth environment, it vanishes entirely. An example is learning to drive on the moon. While most readers will have no prior experience with this task and will be highly uncertain about it initially, we can master it through practice, i.e., by collecting more training data.

Aleatoric uncertainty, on the other hand, captures the inherent uncertainty or randomness in the environment. In the POMDP setting discussed earlier, this is the uncertainty arising from the noise in the observations distribution $p(\mathbf{o}_t|\mathbf{z}_t)$ and the dynamics $p(\mathbf{z}_{t+1}|\mathbf{z}_t, \mathbf{a}_t)$. It cannot be reduced by collecting more training data and will not vanish even if the agent has perfect knowledge of the environment. However, aleatoric uncertainty is still reducable within a single episode of the agent interacting with the environment through active information-gathering behavior. Here, the example is driving through a busy street in a city. No matter how much experience we have, we can never perfectly predict the behavior of other drivers and road users and need to adapt our driving accordingly.

As outlined above, it is common wisdom in the MBRL literature that for successful model-based RL, we need to learn models that capture and address both of these two sources of uncertainty. For instance, Probabilistic Inference for Learning Control (PILCO) (Deisenroth and Rasmussen, 2011) utilizes Gaussian Processes (GPs) that inherently capture both epistemic and aleatoric uncertainty. To integrate more modern deep learning approaches into MBRL Gal and Rasmussen (2016) have extended PILCO using Bayesian neural networks, building on Monte Carlo Dropout (Gal and Ghahramani, 2016). Another highly scalable and popular is the use of probabilistic ensembles (Chua et al., 2018). For these, each member of the ensemble captures aleatoric uncertainty. In contrast, the ensemble as a whole can be viewed as approximating a Bayesian posterior over the model parameters (Lakshminarayanan et al., 2017), thus capturing epistemic uncertainty. After Chua et al. (2018) originally proposed using a simple planning approach on top of such a probabilistic ensemble, many approaches extended either modeling and behavior learning while keeping the underlying ideas of ensembling multiple probabilistic models (Janner et al., 2019; Sekar et al., 2020).

Chapter 5 analyses the role of epistemic and aleatoric uncertainty in state space-based world models presented in literature and, following the insights from this analysis, introduces a new approach to disentangle these two sources of uncertainty.

2.3.4. Probabilistic World Models for RL under Partial Observability

We can now tie together the previously established concepts of probabilistic modeling (Section 2.1), model learning with VI (Section 2.2.2), and RL to formulate a framework for learning behavior with probabilistic world models. This framework will serve as the foundation for the second part of this thesis, where we will present three different approaches to learning behavior with probabilistic world models in Chapter 5, Chapter 6, and Chapter 7.

For our purposes, we will consider a probabilistic world model based on an SSM, which captures the dynamics of the environment and the observation generation process. Additionally, as we aim to learn goal-driven behavior through RL, we need to model the POMDP's reward, which we do using a reward model $p(r_t|z_t)$. Finally, we assume that we have an efficient inference mechanism for the world model, which allows us to approximate the belief state needed for RL in POMDPs. In our case, this inference mechanism is typically based on a Variational Inference (VI) model, which is trained simultaneously with the world model. Figure 2.3 provides a schematic overview of the components of a probabilistic world model and how they relate to the RL problem.

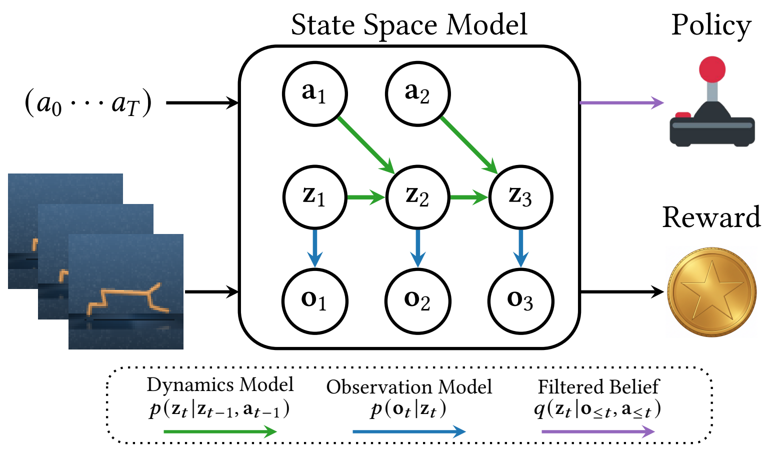


Figure 2.3.: Overview of State Space Models as World Models for RL under partial observability. Given observations and actions, the world model learns both the generative dynamics $p(\mathbf{z}_{t+1}|\mathbf{z}_t, \mathbf{a}_t)$ and the observation generation process $p(\mathbf{o}_t|\mathbf{z}_t)$, as well as an variational distribution $q(\mathbf{z}_t|\mathbf{o}_{\leq t}, \mathbf{a}_{\leq t-1})$ from data collected by the agent. These models, together with a reward model $p(r_t|\mathbf{z}_t)$, allow learning behavior based on the belief extracted by the inference model. For behavior learning we can either directly provide the belief to a model-free algorithm, such as SAC, or use the generative parts of the world model to rollout action in imagination for a model-based approach.

Given such a probabilistic world model, we primarily consider three different approaches to learning the actual behavior, which are described below. For all three approaches, we learn the world model and behavior concurrently in an online fashion, meaning we train the world model on data collected by the agent in the environment.

2.3.4.1. Learning Behavior with Probabilistic World Models

We can use the beliefs extracted from the probabilistic world model for either model-free RL or model-based RL. In this context, model-free approaches still utilize the model to compute an approximate belief state, but then learn a policy directly from this belief state without using on the model itself. In contrast, model-based approaches do not only use the belief state but also the world model's latent state dynamics to explicitly roll out trajectories and predict the effects of actions.

In Chapter 6 and Chapter 7 we consider a model-free approach based on SAC (Haarnoja et al., 2018) (Section 2.3.1.4). Here, the mean of the belief state serves as input to both the actor and critic networks. The policy is learned directly from real-world experience collected in the environment, and the world model only provides a compact and informative state representation.

Additionally, we consider two model-based approaches. The first approach is simple planning in the latent space of the world model using the Cross-Entropy Method (CEM) (Rubinstein, 1997), which we will utilize in Chapter 5. Following Hafner et al. (2019), we use the world model to first generate the belief state and then use its dynamics to simulate several multi-step rollouts and their returns with actions sampled from a Gaussian distribution.

The Gaussian distribution is then updated on the k best-performing action sequences using maximum likelihood, and the process is repeated several times. However, to achieve optimal behavior with this approach, we need to replan the entire action sequence at every time step to account for model errors and noise, which is computationally very expensive.

The second, more prominent, model-based approach is known as latent imagination (Hafner et al., 2020). Instead of planning at each time step, latent imagination learns a parametric policy mapping from the belief state to a distribution over actions. However, unlike SAC, we still roll out the policy in the latent space, predict returns, and train the actor by back-propagating through the world model. To avoid very long rollouts or learning a myopic policy, Hafner et al. (2020) propose learning a value function in the latent space as well. This value function and the predicted rewards are then combined, akin to generalized advantage estimation (GAE) (Schulman et al., 2015b), to trade off bias and variance and serve as target values $v(\mathbf{z}_t)$ for the actor. Given these targets, we can train the policy by Maximizing

$$\mathbb{E}_{q(\mathbf{z}_t|\mathbf{o}_{\leq t},\mathbf{a}_{\leq t-1}) \prod_{k=t}^{t+H} \pi(\mathbf{a}_k|\mathbf{z}_k) p(\mathbf{z}_{k+1}|\mathbf{z}_k,\mathbf{a}_k)} \left[\sum_{k=t}^{t+H} v(\mathbf{z}_k) \right].$$

This approach exploits the fact that the world model's latent dynamics are differentiable, enabling us to propagate gradients through the entire process of simulating and assessing trajectories. Such dynamic backpropagation through the model stands in contrast to model-free actor-critic methods, which can only propagate gradients through the immediate Q-value function and not the environment's dynamics. We will consider this approach in Chapter 5 and Chapter 7.

The choice between using SAC on the belief state and latent imagination involves a trade-off between robustness and sample efficiency. While the latent imagination approach can be more sample-efficient, it also relies significantly on the accuracy of the world model and its ability to accurately predict multi-step rollouts. Conversely, for SAC, the behavior is still trained on real-world data from the replay buffer, and only the belief state is computed using the world model. Crucially, this approach, in principle, still works with world models that lack prediction capability, as long as they can compute concise and effective belief states.

2.4. Distribution Matching to Learn Behavior by Imitation

Imitation Learning (IL) is a powerful alternative to the previously discussed RL paradigm for learning behaviors. For many real-world tasks, demonstrations from a (human) expert performing the task are available or easy to obtain. We can use such demonstrations to learn a policy by imitation, which can be a more straightforward alternative to defining a reward function. While this approach typically cannot surpass the expert's performance, it

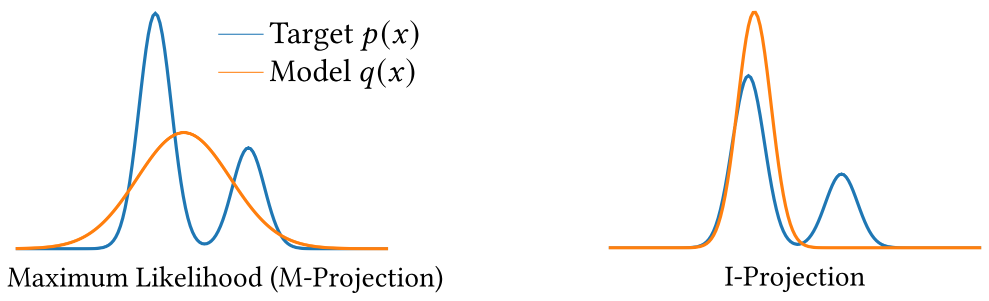


Figure 2.4.: Example of matching a simple Gaussian model to a target distribution with two modes with different objectives. **Left:** The maximum likelihood solution, which is equivalent to the M-Projection, covers both modes of the target. However, to do so, it needs to put significant density in the region between the two modes, which is not supported by the target distribution. **Right:** The I-Projection solution allocates all density to the larger mode of the target distribution, ignoring the smaller mode, while staying within the target distribution's support.

can often help to learn strong initial behavior that we can then refine with Reinforcement Learning (Rajeswaran et al., 2018; Ouyang et al., 2022).

Fundamentally, IL can be framed as a problem of distribution matching, i.e., learning a policy whose behavior distribution matches that of the expert. This perspective connects IL to the previously discussed probabilistic modeling and learning techniques. To capture the inherently multimodal nature of human behavior, we can use the probabilistic mixture models discussed in Section 2.1.2 to represent the behavior distribution. However, simple supervised learning via maximum likelihood estimation of the actions given a state is often insufficient for IL. This approach can lead to issues such as compounding errors (Ross et al., 2011) and may fail to safely capture the multimodal nature of human expert behavior. To learn safe and effective behaviors from multimodal distributions, we need therefore consider more advanced distribution matching methods. In this section, we will derive the foundations for such methods, starting from a Variational Inference perspective. In Chapter 3 we will follow the same recipe to obtain a novel approach for safe IL from multimodal data by modifying the Variational Inference by Policy Search (VIPS) approach (Arenz et al., 2018) (Section 2.2.4.1).

2.4.1. Objectives for Distribution Matching

We already discussed one objective for distribution matching as an alternative to the standard maximum likelihood approach, namely, the KL objective we used for VI (Equation 2.8),

$$\arg\min_{q(x)} \mathrm{KL}\left(q(x)||p(x)\right) \quad \text{with} \quad \mathrm{KL}\left(q(x)||p(x)\right) = \int q(x) \log \frac{q(x)}{p(x)} dx.$$

This objective is also known as the Information Projection (I-Projection). However, the KL is not symmetric, and we obtain an alternative by exchanging the positions of the model

$q(x)$ and the target distribution $p(x)$. The resulting objective is known as the Moment Projection (M-Projection) and is, in fact, equivalent to the standard maximum likelihood

$$\begin{aligned}\arg\min_{q(x)} \mathrm{KL}\left(p(x)||q(x)\right) &= \arg\min_{q(x)} \int p(x) \log \frac{p(x)}{q(x)} dx \\ &= \arg\min_{q(x)} c - \int p(x) \log q(x) dx = \arg\max_{q(x)} \mathbb{E}_{p(x)}[\log q(x)]. \end{aligned} \tag{2.20}$$

Crucially, the choice of objective can have a significant impact on the learned model. For example, the M-Projection, as the name suggests, matches the moments of the model $q(x)$ to the moments of the target distribution $p(x)$. It forces the model to allocate density to all regions where the target distribution has density, without penalizing the model for assigning density to regions where the target distribution has no density. In contrast, the I-Projection has a zero forcing effect, meaning it will force the model to have low density in regions where the target distribution has low density but not penalize the model for ignoring regions where the target distribution has high density. This insight is the key motivation behind using the I-Projection for the safe IL approach introduced in Chapter 3.

We can obtain further distribution matching objectives by using other divergences. One generalization of the KL-divergence is the concept of f-divergences (Ali and Silvey, 1966), which gives rise to a class of objectives of the form

$$\arg\min_{q(x)} \int p(x) f\left(\frac{q(x)}{p(x)}\right) dx \tag{2.21}$$

where f is a convex function with $f(1) = 0$. This class of objectives includes both the I-Projection and the M-Projection, but also many other objectives such as the Jensen-Shannon divergence.

2.4.2. From Variational Inference to Distribution Matching

In the VI setting, we assumed we could not sample from the posterior $p(x)$ but have access to its unnormalized likelihood function. For distribution matching from samples, we will now change this assumption and assume we have no information about the distribution's (unnormalized) likelihood, but are given a fixed set of samples from it. This change results in a supervised learning approach applicable to IL, where the samples are the expert demonstrations.

However, under these assumptions, we cannot directly use most of the previously discussed objectives, as we cannot evaluate $p(x)$ inside the convex function f in Equation 2.21. This limitation explains the predominance of the standard maximum likelihood objective (Equation 2.20), where this term naturally vanishes, and $p(x)$ only occurs in the expectation, which is approximated using the given samples (Equation 2.20).

However, we can still use objectives such as the I-Projection or any other of the f-divergence objectives introduced in Section 2.4.1 by estimating the density ratio $\frac{q(x)}{p(x)}$ from samples. This insight is the key behind the popular class of Generative Adversarial Networks (GANs) (Goodfellow et al., 2014), but also crucial for the approach introduced in Chapter 3. We will now first discuss how to estimate this density ratio from samples and then how to use it for distribution matching with GANs.

2.4.2.1. Density Ratio Estimation

As the name suggests, Density Ratio Estimation (DRE) aims to estimate the ratio between two probability densities $p_1(x)$ and $p_2(x)$, i.e., $r(x) = \frac{p_1(x)}{p_2(x)}$. The key insight here is that estimating this ratio is fundamentally simpler than estimating the individual densities, as it avoids the normalization constants of the distributions.

The most intuitive and common method for DRE is probabilistic classification. If we assign samples from $p_1(x)$ a label of 0 and samples from $p_2(x)$ a label of 1, we can rewrite the density ratio as

$$r(x) = \frac{p_1(x)}{p_2(x)} = \frac{p(x|y=1)}{p(x|y=0)} = \frac{p(y=0|x)p(x)}{p(y=0)} \frac{p(y=1)}{p(y=1|x)p(x)} = \frac{p(y=0)}{p(y=1)} \cdot \frac{p(y=1|x)}{p(y=0|x)}.$$

Here, crucially, the fraction simplifies by removing $p(x)$, which removes the normalization constant and only leaves us with the task of estimating the conditional probabilities $p(y=1|x)$ and $p(y=0|x)$, which is a standard classification problem. More specifically, if we train a logistic regressor $C(x)$ to predict the probability $C(x) = p(y=1|x)$ using the standard binary-cross-entropy objective, the logits $\phi(x) = \text{sigmoid}^{-1}(C(x))$ of this classifier are equivalent to the log of the density ratio. To realize this connection, we first need to realize that under these assumptions, $C(x)$ is optimal for $C(x) = q(x)/\big(q(x) + p(x)\big)$ (Bishop, 2006). Using this, we can rewrite the log density ratio as

$$\log \frac{q(x)}{p(x)} = \log \frac{q(x)/\big(q(x)+p(x)\big)}{p(x)/\big(q(x)+p(x)\big)} = \log \frac{C(x)}{1-C(x)} = \text{sigmoid}^{-1}\big(C(x)\big) = \phi(x).$$

Given this, we can now use the entire toolbox for standard logistic regression to estimate our density ratio instead of learning two separate fully generative models for $p_1(x)$ and $p_2(x)$.

We will use this exact method of density ratio estimation in Chapter 3. However, to fully understand the relation to GANs, we also need to consider a more general framework for density ratio estimation based on Bregman divergences (Sugiyama et al., 2012). Such Bregman divergences (Bregman, 1967) are measures of distance between elements of convex sets, defined by a strictly convex function $f(t)$. The framework of Sugiyama et al. (2012) builds on the inside that we can use arbitrary Bregman divergences to learn density ratio estimators. Similar to how using different f's can lead to different models when matching two distributions by minimizing an f-divergence, different choices of f for the Bregman divergence can lead to different density ratio estimators.

2.4.2.2. Generative Adversarial Networks

The intuition behind Generative Adversarial Networks (GANs) can be understood as a competition between a forger and a detective. In this example, the forger tries to create counterfeits that fool the detective while the detective tries to detect the counterfeits. Here, both parties need to continuously improve their strategies to outsmart each other until the forger's counterfeits are indistinguishable from the real thing.

Formally, as introduced by Goodfellow et al. (2014), GANs consist of two neural networks trained by playing a two-player minimax game. The first network, the generator, is a latent variable model $q(x) = \int q(x|z)q(z)dz$ that generates samples x. Here, usually, $q(z)$ is a simple isotropic Gaussian distribution, while $q(x|z)$ is a Dirac delta distribution parameterized by an arbitrary neural network that maps the latent variable z to the data space. Under these assumptions, the density of the generator $q(x)$ is itself intractable, and the capability of using models with intractable densities is one of the key advantages of GANs over prior methods. The second network, the discriminator $C(x)$, predicts the probability that a given sample x is from the target distribution $p(x)$ and not generated by $q(x)$. Goodfellow et al. (2014) suggest training the networks by optimizing

$$\arg\min_{q(x)} \arg\max_{C(x)} \mathbb{E}_{p(x)}\left[\log C(x)\right] + \mathbb{E}_{q(x)}\left[\log(1 - C(x))\right]. \tag{2.22}$$

and prove that $q(x)$ will minimize the Jensen-Shannon divergence between $p(x)$ and $q(x)$ at convergence.

The connection with the estimation of the density ratio becomes apparent when one realizes that the resulting discriminator $C(x)$ is a logistic regression model trained to predict the probability that a given sample x is from the target distribution $p(x)$ using the binary cross-entropy loss. Thus, as discussed in the previous section, its logits approximate the log density ratio $\log \frac{q(x)}{p(x)}$.

Building on this relation or related insights, several works (Nowozin et al., 2016; Uehara et al., 2016; Poole et al., 2016), present versions of the GAN that minimize arbitrary f-divergences between the model distribution $q(x)$ and the target distribution $p(x)$. In particular, Uehara et al. (2016) make the connection to density ratio estimation explicit by using the Bregman divergence framework of Sugiyama et al. (2012) to derive their approach. We will further examine the relations between these approaches with a focus on the I-Projection in Chapter 3.

2.4.3. Distribution Matching for Imitation Learning

The advanced distribution matching concepts introduced in Section 2.4.2 allow us to go beyond simple supervised learning by maximum likelihood and design IL approaches along two main axes. First, we can decide which target distribution to match, and second, we can choose the objective for matching.

Regarding the target distribution, the most direct approach is known as Behavioral Cloning (BC) (Pomerleau, 1991; Osa et al., 2018). BC aims to match the expert's conditional action policy $\pi(a|s)$, i.e., the distribution of actions the expert took in each state s. The crucial limitation of this approach is that any error in the learned policy will compound when it is executed. Because the policy is trained only on states visited by the expert, any mistake can lead the agent to out-of-distribution regions of the state space, where optimal actions are not known. A more robust, but harder-to-realize, alternative is to match the expert's state-action distribution $p(s, a)$ or its marginal state distribution $p(s)$. Intuitively, these force the agent to both be in the same states and take the same actions as the expert or only be in the same states as the expert, respectively. The main problem with these approaches is that evaluating the state-action distribution $p(s, a)$ or the marginal state distribution $p(s)$ requires knowledge of the system dynamics, which is usually not available. However, if we can sample from the system dynamics, approaches such as Generative Adversarial Imitation Learning (GAIL) (Ho and Ermon, 2016) can still match these distributions by using the GAN framework introduced in Section 2.4.2.

The second choice, which objective to use, is more prominently featured in this thesis, This choice determines the nature of the learned policy, especially when dealing with the versatile and multimodal behavior often demonstrated by humans. As introduced in Section 2.4.1, the maximum likelihood objective used in standard BC is mode-covering. It forces the learned model to assign probability mass everywhere the expert distribution has mass. When the model has limited capacity and the expert is multimodal, this property leads to disastrous mode-averaging. The learned policy averages the distinct expert behaviors, resulting in potentially unsafe actions. For example, if an obstacle has to be avoided on the left or the right, averaging between these options would lead to colliding with the obstacle head-on. In contrast, the reverse KL divergence, which defines the Information Projection (I-Projection), is mode-seeking or zero-forcing. It penalizes the learner for assigning probability mass where the expert has none. This allows a capacity-limited model to ignore some modes of the expert's behavior and focus its entire capacity on accurately representing a single, valid mode. In the obstacle avoidance example, a policy trained with the I-Projection would learn to go either left or right but not to average them into a collision. This property is crucial for safely imitating complex human behavior. Symmetric objectives, such as the Jensen-Shannon divergence used in GAIL, represent a compromise but still exhibit mode-covering tendencies and do not fully resolve the safety concerns of mode-averaging.

Motivated by the beneficial properties of the I-Projection for IL Chapter 3 introduces a novel, practical, robust, and efficient approach to optimize it based on samples. Furthermore, we provide empirical results that demonstrate the advantages of this approach over prior methods for IL.

3. Expected Information Maximization: Using the I-Projection for Mixture Density Estimation

As discussed in the introduction and Section 2.4, learning from human behavior presents a powerful framework for training embodied agents. However, for this approach to succeed, the agent must be able to learn from the diverse human behavior safely. Failing to represent this multimodality appropriately can lead to policies that are not only suboptimal but also unsafe. In this chapter, we introduce a novel approach to address this challenge by answering the first research question of this thesis:

Q1 How can we learn multimodal probabilistic models that focus on modes they can represent and avoid averaging?

Our solution is a novel approach called Expected Information Maximization (EIM), designed to learn latent variable models, particularly mixture models, from demonstration data. Inspired by previous work on Variational Inference for Mixture Models (Arenz et al., 2018)(Section 2.2.4.1), EIM avoids using the likelihood as an objective and instead uses the Information Projection. This approach allows the model to focus on the data it can represent, avoiding the averaging effects that can lead to suboptimal or unsafe policies. While EIM shares the need for density ratio estimation with adversarial methods, it is not adversarial itself, as the density ratio is between two fixed distributions. This results in an optimization procedure more akin to the original Expectation Maximization (Dempster et al., 1977) approach, where a bound is iteratively optimized and tightened.

While EIM introduces a general learning approach, its primary motivation and focus is on learning multimodal behavior by imitation. Further development of EIM, including an application to real-world robotic imitation, is summarized in Section 3.7.

The following was published as Expected Information Maximization: Using the I-Projection for Mixture Density Estimation *(Philipp Becker, Oleg Arenz, and Gerhard Neumann, 2020) in the 8th International Conference on Learning Representations, ICLR 2020. Reprinted with permission of the authors. Wording, notation, and formulations were revised in several places. The contents of the original publication were restructured. In particular, the content of the preliminary section of the original publication was integrated into Section 2.4. Section 3.7 briefly summarizes several follow-up papers I contributed to during my PhD studies.*

The work builds on my Master Thesis with the same title but extends it with additional discussions and experiments for Expected Information Maximization with Marginal Distributions.

Additionally, while the lower bound for the conditional case was already included in the thesis, I derived, implemented, and evaluated the realization of Expected Information Maximization for Nonlinear Mixtures of Experts afterward as part of my PhD studies.

3.1. Introduction

Learning the density of highly multimodal distributions is a challenging machine learning problem relevant to many fields such as modeling human behavior (Pentland and Liu, 1999). Most common methods rely on maximizing the likelihood of the data. It is well known that the maximum likelihood solution corresponds to computing the M(oment)-projection of the data distribution to the parametric model distribution (Bishop, 2006). Yet, the M-Projection averages over multiple modes in case the model distribution is not rich enough to fully represent the data (Bishop, 2006). This averaging effect can result in poor models, that put most of the probability mass in areas that are not covered by the data. The counterpart of the M-Projection is the I(nformation)-projection. The I-Projection concentrates on the modes the model is able to represent and ignores the remaining ones. Hence, it does not suffer from the averaging effect (Bishop, 2006).

In this paper, we explore the I-Projection for mixture models which are typically trained by maximizing the likelihood via Expectation Maximization (EM) (Dempster et al., 1977). Despite the richness of mixture models, the averaging problem remains as we typically do not know the correct number of modes and it is hard to identify all modes of the data correctly. By the use of the I-Projection, our mixture models do not suffer from this averaging effect and can generate more realistic samples that are less distinguishable from the data. In this paper we concentrate on learning Gaussian mixture models and conditional Gaussian mixtures of experts (Jacobs et al., 1991) where the mean and covariance matrix are generated by deep neural networks.

We propose Expected Information Maximization (EIM), a novel approach capable of computing the I-Projection between the model and the data. By exploiting the structure of the I-Projection, we can derive a variational upper bound objective, which was previously used in the context of Variational Inference (Maaløe et al., 2016; Ranganath et al., 2016; Arenz et al., 2018). In order to work with this upper bound objective based on samples, we use a discriminator to approximate the required density ratio, relating our approach to GANs (Goodfellow et al., 2014; Nowozin et al., 2016; Uehara et al., 2016). The discriminator also allows us to use additional discriminative features to improve model quality. In our experiments, we demonstrate that EIM is much more effective in computing the I-Projection than recent GAN approaches. We apply EIM to a synthetic obstacle avoidance task, an inverse kinematic task of a redundant robot arm as well as a pedestrian and car prediction task using the Stanford Drone Dataset (Robicquet et al., 2016) and a traffic dataset from the Next Generation Simulation program (U.S. Dep. of Transportation, 2016).

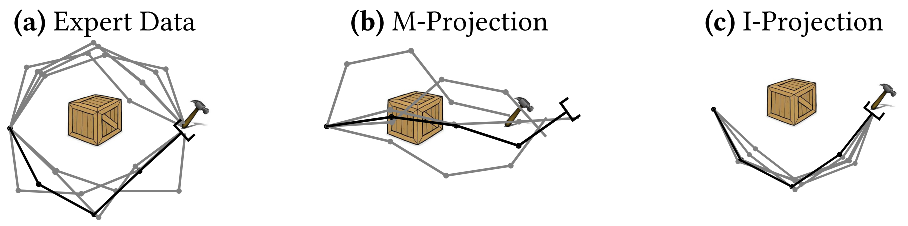


Figure 3.1.: Illustration of the I-Projection vs. the M-Projection for modeling behavior. **(a)**: A robot reaches a target point while avoiding an obstacle. There are two different types of solutions, above and below the obstacle. A single Gaussian is fitted to the expert data in joint space. **(b)**: The M-Projection fails to reach the target and collides with the obstacle. **(c)**: The I-Projection ignores the second mode and reaches the target while avoiding the obstacle.

3.2. Related Work

We will now discuss competing methods for computing the I-Projection that are based on GANs. Those are, to the best of our knowledge, the only other approaches capable of computing the I-Projection solely based on samples of the target distribution. Furthermore, we will distinguish our approach from approaches based on Variational Inference (VI) that also use the I-Projection.

Variational Inference. The I-Projection is a common objective in VI (Opper and Saad, 2001; Bishop, 2006; Kingma and Welling, 2013). Those methods aim to fit tractable approximations to intractable distributions of which the unnormalized density is available. EIM, on the other hand, does not assume access to the unnormalized density of the target distributions but only to samples. Hence, it is not a VI approach, but a density estimation approach. However, our approach uses an upper bound that has been previously applied to VI (Maaløe et al., 2016; Ranganath et al., 2016; Arenz et al., 2018). EIM is especially related to the VIPS algorithm (Arenz et al., 2018), which we extend from the VI case to the density estimation case. Additionally, we introduce conditional latent variable models into the approach.

Generative Adversarial Networks. While the original GAN approach minimizes the Jensen-Shannon Divergence (Goodfellow et al., 2014), GANs have since been adapted to a variety of other distance measures between distributions, such as the Wasserstein distance (Arjovsky et al., 2017), symmetric KL (Chen et al., 2018) and arbitrary f-divergences (Ali and Silvey, 1966; Nowozin et al., 2016; Uehara et al., 2016; Poole et al., 2016). Since the I-Projection is a special case of an f-divergence, those approaches are of particular relevance to our work. Nowozin et al. (2016) use a variational bound for f-divergences (Nguyen et al., 2010) to derive their approach, the f-GAN. Uehara et al. (2016) use a bound that directly follows from the density ratio estimation under Bregman divergences framework introduced by Sugiyama et al. (2012) to obtain their b-GAN. While the b-GAN's discriminator directly estimates the density ratio, the f-GAN's discriminator estimates an invertible mapping of the density ratio. Yet, in the case of the I-Projection, both the f-GAN and the b-GAN yield the same objective, as we show in Appendix A.3. For both the f-GAN and b-GAN the desired f-divergence determines the discriminator objective. Uehara et al. (2016) note that the discriminator objective, implied by the I-Projection, is unstable. As both approaches are formulated in a general way to minimize any f-divergence, they do not exploit the special structure of the I-Projection. Exploiting this structure permits us to apply a tight upper bound of the I-Projection for latent variable models, which results in a higher quality of the estimated models.

Li et al. (2019) introduce an adversarial approach to compute the I-Projection based on density ratios, estimated by logistic regression. Yet, their approach assumes access to the unnormalized target density, i.e., they are working in a VI setting. The most important difference to GANs is that we do not base EIM on an adversarial formulation and no adversarial game has to be solved. This removes a major source of instability in the training process, which we discuss in more detail in Section 3.4.1.

3.3. Expected Information Maximization

Expected Information Maximization (EIM) is a general algorithm for minimizing the I-Projection for any latent variable model. We first derive EIM for general marginal latent variable models, i.e., $q(x) = \int q(x|z)q(z)dz$ and subsequently extend our derivations to conditional latent variable models, i.e., $q(x|y) = \int q(x|z,y)q(z|y)dz$. EIM uses an upper bound for the objective of the marginal distribution. Similar to Expectation-Maximization (EM), our algorithm iterates between an M-step and an E-step. In the corresponding M-step, we minimize the upper bound and in the E-step we tighten it using a variational distribution.

3.3.1. EIM for General Latent Variable Models

The I-Projection can be simplified using a (tight) variational upper bound (Arenz et al., 2018) which can be obtained by introducing an auxiliary distribution $\tilde{q}(z|x)$ and using Bayes rule

$$\mathrm{KL}\left(q(x)||p(x)\right) = \underbrace{U_{\tilde{q},p}(q)}_{\text{upper bound}} \underbrace{- \mathbb{E}_{q(x)}\left[\mathrm{KL}\left(q(z|x)||\tilde{q}(z|x)\right)\right]}_{\geq 0},$$

where

$$U_{\tilde{q},p}(q) = \iint q(x|z)q(z)\left(\log \frac{q(x|z)q(z)}{p(x)} - \log \tilde{q}(z|x)\right) dzdx. \tag{3.1}$$

The derivation of the bound is given in Appendix A.1. It is easy to see that $U_{\tilde{q},p}(q)$ is an upper bound as the expected KL term is always non-negative. In the corresponding E-step, the model from the previous iteration, which we denote as $q_t(x)$, is used to tighten the bound by setting $\tilde{q}(z|x) = q_t(x|z)q_t(z)/q_t(x)$. In the M-step, we update the model distribution by minimizing the upper bound $U_{\tilde{q},p}(q)$. Yet, opposed to Arenz et al. (2018), we cannot work directly with the upper bound since it still depends on $\log p(x)$, which we cannot evaluate. However, we can reformulate the upper bound by setting the given relation for $\tilde{q}(z|x)$ of the E-step into Equation 3.1,

$$U_{q_t,p}(q) = \int q(z) \left(\int q(x|z) \log \frac{q_t(x)}{p(x)} dx + \mathrm{KL}\left(q(x|z)||q_t(x|z)\right) \right) dz + \mathrm{KL}\left(q(z)||q_t(z)\right). \tag{3.2}$$

The upper bound now contains a density ratio between the old model distribution and the data. This density ratio can be estimated using samples of q_t and p, for example, by using logistic regression as shown in Section 2.4.2.1. We can use the logits $\phi(x)$ of such a logistic regressor to estimate the log density ratio $\log(q_t(x)/p(x))$ in Equation 3.2. This yields an upper bound

$$U_{q_t,\phi}(q) = \int q(z) \left(\int q(x|z)\phi(x) dx + \mathrm{KL}\left(q(x|z)||q_t(x|z)\right) \right) dz + \mathrm{KL}\left(q(z)||q_t(z)\right). \tag{3.3}$$

that depends on $\phi(x)$ instead of $p(x)$. Optimizing this bound corresponds to the M-step of our approach. In the E-step, we set q_t to the newly obtained q and retrain the density ratio estimator $\phi(x)$. Both steps formally result in the following bilevel optimization problem

$$q_{t+1} \in \arg\min_{q(x)} U_{q_t,\phi^*}(q) \quad \text{s.t.} \quad \phi^*(x) \in \arg\min_{\phi(x)} \mathrm{BCE}(\phi(x), p(x), q_t(x)).$$

Using a discriminator also comes with the advantage that we can use additional discriminative features $g(x)$ as input to our discriminator that are not directly available for the generator. For example, if x models trajectories of pedestrians, $g(x)$ could indicate whether the trajectory reaches any positions that are not plausible such as rooftops or trees. These features simplify the discrimination task and can therefore improve our model accuracy which is not possible with M-Projection based algorithms such as EM.

3.3.2. EIM for General Conditional Latent Variable Models

For conditional distributions, we aim at finding the conditional I-Projection

$$\arg\min_{q(x|y)} \mathbb{E}_{p(y)}\left[\mathrm{KL}\left(q(x|y)||p(x|y)\right)\right].$$

The derivations for the conditional upper bound follow the same steps as the derivations in the marginal case, where all distributions are extended by the context variable y. We refer to the supplement for details. The log density ratio estimator $\phi(x, y)$ now discriminates between samples of the joint distribution of x and y. For training $\phi(x, y)$ we generate a new sample x for each context y, using the distribution $q_{\text{old}}(x|y)$. Hence, as the context distribution is the same for the true data and the generated data, the log density ratio of the conditional distributions is equal to the log density ratio of the joint distributions.

3.3.3. EIM for Gaussian Mixtures Models

We consider Gaussian mixture models with D components, i.e., multivariate Gaussian distributions $q(\mathbf{x}|z_i) = \mathcal{N}(\boldsymbol{\mu}_i, \Sigma_i)$ and a categorical distribution $q(z) = \mathrm{Cat}(\boldsymbol{\pi})$ for the coefficients. As the latent distribution $q(z)$ is discrete, the upper bound in EIM (Equation 3.2) simplifies, as the integral over z can be written as a sum. Similar to the EM-algorithm, this objective can be updated individually for the coefficients and the components. For both updates, we will use similar update rules as defined in the VIPS algorithm (Arenz et al., 2018). VIPS uses a trust region optimization for the components and the coefficients, where both updates can be solved in closed form as the components are Gaussian. The trust regions prevent the new model from going too far away from q_t where the density ratio estimator is inaccurate, and hence, further stabilize the learning process. We will now sketch both updates and refer to Appendix A.1 for the full details.

For updating the coefficients, we assume that the components have not yet been updated, and therefore $\mathrm{KL}\left(q(\mathbf{x}|z_i)||q_t(\mathbf{x}|z_i)\right) = 0$ for all z_i. The objective for the coefficients thus simplifies to

$$\arg\min_{q(z)} \sum_{i=1}^{D} q(z_i)\phi(z_i) + \mathrm{KL}\left(q(z)||q_t(z)\right) \quad \text{with} \quad \phi(z_i) = \mathbb{E}_{q(\mathbf{x}|z_i)}\left[\phi(\mathbf{x})\right], \tag{3.4}$$

where $\phi(z_i)$ can be approximated using samples from the corresponding component. This objective can easily be optimized in closed form, as shown in the VIPS algorithm (Arenz et al., 2018). We also use a KL trust-region to specify the step size of the update. For updating the individual components, the objective simplifies to

$$\arg\min_{q(\mathbf{x}|z_i)} \mathbb{E}_{q(\mathbf{x}|z_i)}\left[\phi(\mathbf{x})\right] + \mathrm{KL}\left(q(\mathbf{x}|z_i)||q_t(\mathbf{x}|z_i)\right). \tag{3.5}$$

As in VIPS, this optimization problem can be solved in closed form using the MORE algorithm (Abdolmaleki et al., 2015). The MORE algorithm uses a quadratic surrogate function that locally approximates $\phi(\mathbf{x})$. The resulting solution optimizes Equation 3.5 under a KL trust-region. The pseudo-code of EIM for GMMs can be found in Algorithm 1.

3.3.4. EIM for Gaussian Mixtures of Experts

In the conditional case, we consider mixtures of experts consisting of D multivariate Gaussians, whose parameters depend on an input $\mathbf{y}$ in a nonlinear fashion, i.e., $q(\mathbf{x}|z_i, \mathbf{y}) = \mathcal{N}\left(\psi_{\mu,i}(\mathbf{y}), \psi_{\Sigma,i}(\mathbf{y})\right)$ and the gating is given by a neural network with softmax output. We again decompose the resulting upper bound into individual update steps for the components and the gating. Yet, closed-form solutions are no longer available and we need to resort to gradient-based updates. The objective for updating the gating is given by

$$\arg\min_{q(z|\mathbf{y})} \sum_{i=1}^{D} \left(\mathbb{E}_{p(\mathbf{y})q(\mathbf{x}|z_i,\mathbf{y})}\left[q(z_i|\mathbf{y})\phi(\mathbf{x},\mathbf{y})\right]\right) + \mathbb{E}_{p(\mathbf{y})}\left[\mathrm{KL}\left(q(z|\mathbf{y})||q_t(z|\mathbf{y})\right)\right]. \tag{3.6}$$

```
EIM-for-GMMs($\{\mathbf{x}_\mathrm{p}^{(j)}\}_{j=1\cdots N}, q(\mathbf{x})$);
Input: Data $\{\mathbf{x}_\mathrm{p}^{(j)}\}_{j=1\cdots N}$, Initial Model $q(\mathbf{x}) = \sum_{i=1}^{D} q(\mathbf{x}|z_i)q(z_i) = \sum_{i=1}^{D} \pi_i \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_i, \Sigma_i)$
for $i$ in number of iterations do
    E-Step:
    $q_t(z) = q(z)$, $q_t(\mathbf{x}|z_i) = q(\mathbf{x}|z_i)$ for all components $i$
    Update Density Ratio Estimator:
    sample data from model $\{\mathbf{x}_\mathrm{q}^{(j)}\}_{j=1\cdots N} \sim q_t(\mathbf{x})$
    retrain density ratio estimator $\phi(\mathbf{x})$ on $\{\mathbf{x}_\mathrm{p}^{(j)}\}_{j=1\cdots N}$ and $\{\mathbf{x}_\mathrm{q}^{(j)}\}_{j=1\cdots N}$
    M-Step Coefficients:
    for $i$ in number of components do
        compute loss $l_i = \frac{1}{N}\sum_{j=1}^{N} \phi\left(\mathbf{x}_\mathrm{q}^{(j)}\right)$ with samples $\{\mathbf{x}_\mathrm{q}^{(j)}\}_{j=1\cdots N} \sim q_t(\mathbf{x}|z_i)$
    end
    update $q(z)$ using losses $l_i$ and MORE equations
    M-Step Components:
    for $i$ in number of components do
        fit $\hat{\phi}(\mathbf{x})$ surrogate to pairs $\left(\mathbf{x}_\mathrm{q}^{(j)}, \phi\left(\mathbf{x}_\mathrm{q}^{(j)}\right)\right)$ with samples $\{\mathbf{x}_\mathrm{q}^{(j)}\}_{j=1\cdots N} \sim q_t(\mathbf{x}|z_i)$
        update $q(\mathbf{x}|z_i)$ using surrogate $\hat{\phi}(\mathbf{x})$ and MORE equations
    end
end
```

Algorithm 1: Expected Information Maximization for Gaussian Mixture Models.

We minimize this equation w.r.t. the parameters of the gating by gradient descent using the Adam (Kingma and Ba, 2014) algorithm. The objective for updating a single component i is given by

$$\arg\min_{q(\mathbf{x}|z_i,\mathbf{y})} \mathbb{E}_{\tilde{p}(\mathbf{y}|z_i)}\left[\mathbb{E}_{q(\mathbf{x}|z_i,\mathbf{y})}\left[\phi(\mathbf{x},\mathbf{y})\right] + \mathrm{KL}\left(q(\mathbf{x}|z_i,\mathbf{y})||q_t(\mathbf{x}|z_i,\mathbf{y})\right)\right], \tag{3.7}$$

where $\tilde{p}(\mathbf{y}|z_i) = p(\mathbf{y})q(z_i|\mathbf{y})/q(z_i)$. Note that we normalized the objective by $q(z_i) = \int p(\mathbf{y})q(z_i|\mathbf{y})d\mathbf{y}$ to ensure that also components with a low prior $q(z_i)$ get large enough gradients for the updates. As we have access to the derivatives of the density ratio estimator w.r.t. $\mathbf{x}$, we can optimize Equation 3.7 with gradient descent using the reparametrization trick (Kingma and Welling, 2013) and Adam (Kingma and Ba, 2014).

3.4. Qualitative Comparison to Related Generative Approaches

3.4.1. Generative Adversarial Networks

There is a close relation between EIM and GANs due to the use of a logistic discriminator for the density ratio estimation. It is therefore informative to investigate the differences in

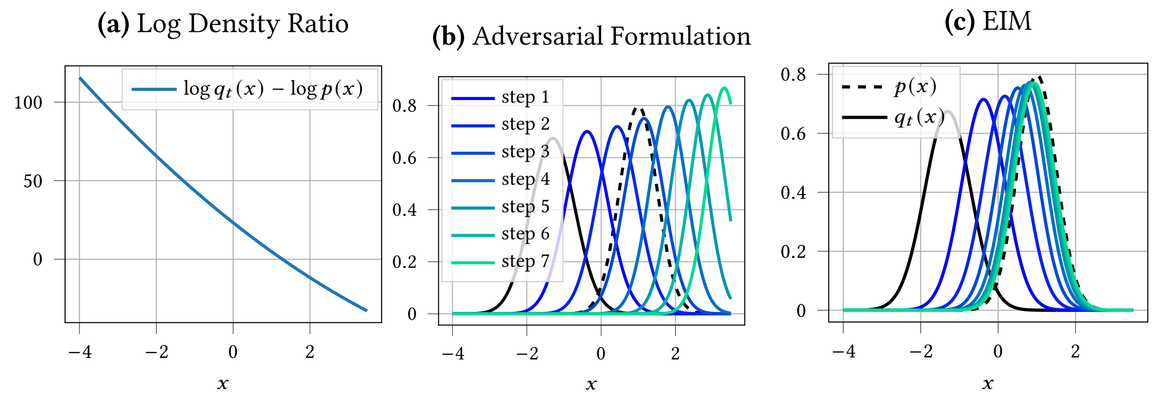


Figure 3.2.: An illustrative example of the benefits of EIM versus an adversarial formulation. **(a)**: The true log density ratio of the model and the target distribution (both are Gaussians). The location of the optimum is unbounded. **(b)**: In the adversarial formulation, the generator minimizes the expected log density ratio. If we neglect that the adversarial discriminator changes with every update step of the generator, the generator updates yield an unbounded solution. Hence, overly aggressive generator updates yield unstable behavior. **(c)**: The upper bound of EIM introduces an additional KL-term as objective. Optimizing this objective directly yields the optimal solution without the need to recompute the density ratio estimate.

the case without latent variables. In an adversarial formulation, the density ratio estimator would directly replace the density ratio in the original I-Projection equation, i.e.,

$$\arg\min_{q(x)} \int q(x)\phi^*(x)dx \quad \text{s.t.} \quad \phi^*(x) \in \arg\min_{\phi(x)} \text{BCE}(\phi(x), p(x), q^*(x)).$$

However, such adversarial games are often hard to optimize. In contrast, EIM offers a bilevel optimization problem where the discriminator is explicitly learned on the old data distribution $q_t(x)$,

$$\arg\min_{q(x)} \int q(x)\phi^*(x)dx + \text{KL}\left(q(x)||q_t(x)\right)$$
$$\text{s.t. } \phi^*(x) \in \arg\min_{\phi(x)} \text{BCE}(\phi(x), p(x), q_t(x)).$$

Thus, there is no circular dependency between the optimal generator and the optimal discriminator. Figure 3.2 illustrates that the proposed non-adversarial formulation does not suffer from too large model updates. Choosing the number and step-size of the updates is thus far less critical.

3.4.2. Expectation-Maximization

EIM can also be seen as the counterpart of Expectation-Maximization (EM). While EM optimizes the M-Projection with latent variable models, EIM uses the I-Projection. However, both approaches decompose the corresponding projections into an upper bound (or lower bound for EM) and a KL-term that depends on the conditional distribution $q(z|x)$ to tighten this bound.

Recall that the Expectation-Maximization (EM) algorithm (Dempster et al., 1977) maximizes the log-likelihood of the data by iteratively maximizing and tightening the following lower bound

$$\mathbb{E}_{p(x)}\left[\log q(x)\right] = \mathbb{E}_{p(x)}\left[\int \tilde{q}(z|x)\log\frac{q(x,z)}{\tilde{q}(z|x)}dz\right] + \mathbb{E}_{p(x)}\left[\int \tilde{q}(z|x)\log\frac{\tilde{q}(z|x)}{q(z|x)}dz\right]$$
$$= \underbrace{\mathcal{L}(q,\tilde{q})}_{\text{lower bound}} + \underbrace{\mathbb{E}_{p(x)}\left[\mathrm{KL}\left(\tilde{q}(z|x)||q(z|x)\right)\right]}_{\geq 0}.$$

It is instructive to compare our upper bound (Equation 3.1) to this lower bound. As mentioned, maximizing the likelihood is equivalent to minimizing the M-Projection, i.e., $\arg\min_{q(x)} \mathrm{KL}\left(p(x)||q(x)\right)$, where, in relation to our objective, the model and true distribution have switched places in the non-symmetric KL objective. Like our approach, EM introduces an auxiliary distribution $\tilde{q}(z|x)$ and bounds the objective from below by subtracting the KL between auxiliary distribution and model, i.e., $\mathrm{KL}\left(\tilde{q}(z|x)||q(z|x)\right)$. In contrast, we obtain our upper bound by adding $\mathrm{KL}\left(q(z|x)||\tilde{q}(z|x)\right)$ to the objective. Again, the distributions have exchanged places within the KL.

3.5. Empirical Evaluation

A reference implementation of Expected Information Maximization for Gaussian Mixtures and Mixtures of Experts is available at:

```
https://github.com/pbecker93/ExpectedInformationMaximization
```

We compare our approach to GANs and perform an ablation study on a toy task, with data sampled from known mixture models. We further apply our approach to two synthetic datasets, learning the joint configurations of a planar robot as well as a non-linear obstacle avoidance task, and two real datasets, namely the Stanford Drone Dataset (Robicquet et al., 2016) and a traffic dataset from the Next Generation Simulation program. A full overview of all hyperparameters and network architectures can be found in Appendix A.2.

3.5.1. Comparison to Generative Adversarial Approaches and Ablation Study

We compare to the f-GAN which is the only other method capable of minimizing the I-Projection solely based on samples. We use data sampled from randomly generated GMMs with different numbers of components and dimensionalities. To study the influence of the previously mentioned differences of EIM to generative adversarial approaches, we also perform an ablation study. We compare to a version of EIM where we neglect the additional KL-term (EIM, no KL), a version where we trained all components and the

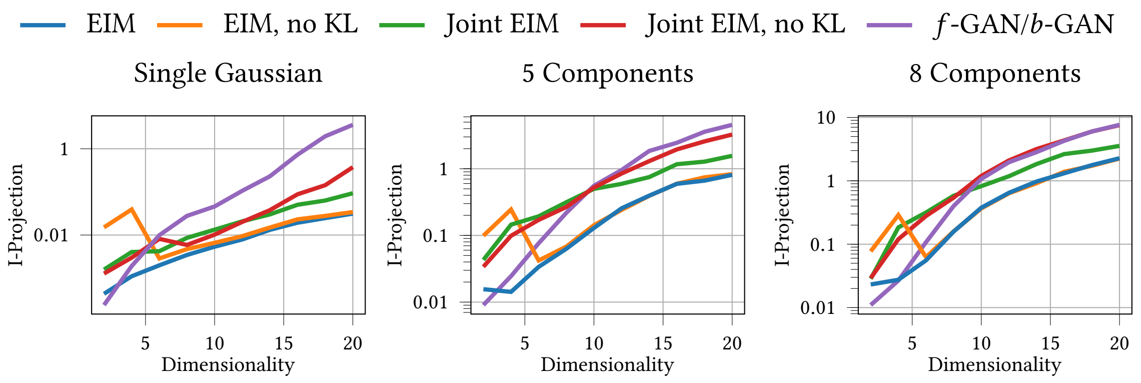


Figure 3.3.: Average I-Projection achieved for EIM, the f-GAN, and the modified EIM versions. The task is to fit a model to samples from a randomly generated GMM of different dimensions. Both the model and the target GMM have the same number of components. EIM clearly outperforms the generative adversarial approaches, especially for larger dimensions. The ablation study shows that the separated, closed-form updates yield better results. Neglecting the KL has a big influence on lower dimensional models, but is outweighed by the error of the discriminator at higher dimensions.

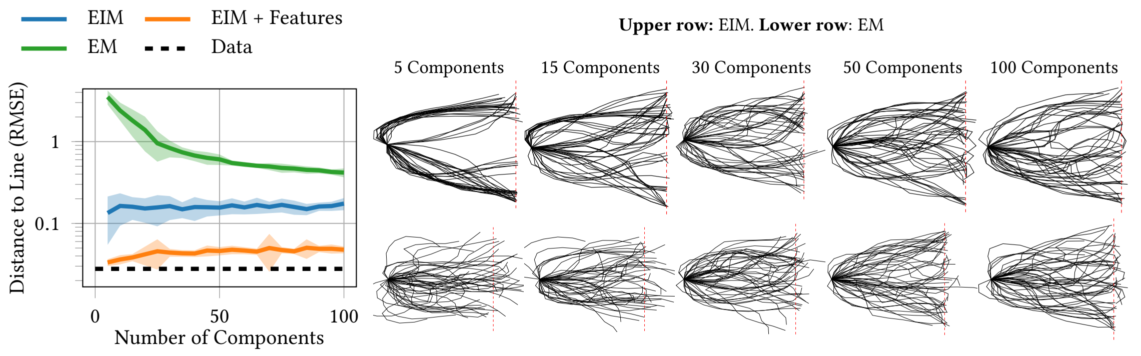


Figure 3.4.: Average distance to line (left) and samples (right) for robot line reaching. While EIM for small numbers of components ignores modes, not considering the whole line, it learns models that achieve the underlying task, i.e., reach the line. Providing additional information to the density ratio estimator further decreases the average distance to the line. EM, on the other hand, averages over the modes, and thus, fails to reach the line even for large numbers of components.

coefficients jointly using gradient descent (Joint EIM), and a version where we do both (Joint EIM, no KL). The average I-Projection achieved by the various approaches can be found in Figure 3.3.

3.5.2. Line Reaching with Planar Robot

We extended the introductory example of the planar reaching task and collected expert data from a 10-link planar robot tasked with reaching a point on a line. We fitted GMMs with an increasing number of components using EIM, EIM with additional features, where the end-effector coordinates for a given joint configuration were provided, and EM. Even for a large number of components, we see effects similar to the introductory example, i.e., the M-Projection solution provided by EM fails to reach the line while EIM manages to

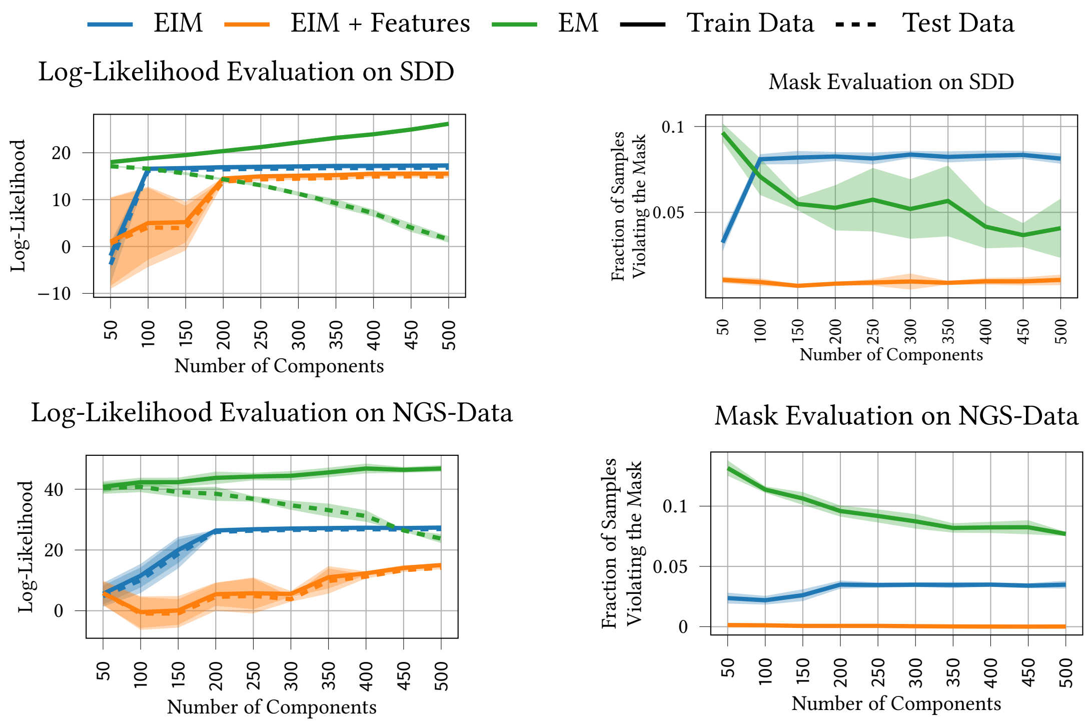


Figure 3.5.: Results on the traffic prediction tasks. Naturally, EM achieves the highest training log-likelihood. Yet, for large numbers of components, a severe amount of overfitting is observed. EIM, on the other hand, has no problems working with high numbers of components and achieves a higher test log-likelihood, despite optimizing a different objective. We also provided 'road masks' as additional features for the discriminator. We used this road mask to evaluate how realistic the generated samples are. While EIM without features produced more realistic samples on the Lankershim dataset, we needed the feature input to outperform EM on this evaluation on the SDD dataset.

do so. For small numbers, EIM ignores parts of the line, while more and more parts of it get covered as we increase the number of components. With the additional features, the imitation of the line reaching was even more accurate. See Figure 3.4, for the average distance between the end-effector and the line as well as samples from both EM and EIM.

3.5.3. Pedestrian and Traffic Prediction

We evaluated our approach on data from the Stanford Drone Dataset (SDD) (Robicquet et al., 2016) and a traffic dataset from the Next Generation Simulation (NGS) program (U.S. Dep. of Transportation, 2016). The SDD data consists of trajectories of pedestrians, bikes, and cars and we only used the data corresponding to a single video of a single scene (Video 1, deathCircle). The NGS data consists of trajectories of cars and we considered the data recorded on Lankershim Boulevard. In both cases we extracted trajectories of length 5, yielding highly multimodal data due to pedestrians, bikes, and cars moving at different speeds and in different directions. We evaluated on the achieved log-likelihood of EIM and EM, see Figure 3.5. EM achieves the highest likelihood as it directly optimizes this measure. However, we can already see that EM massively overfits when we increase

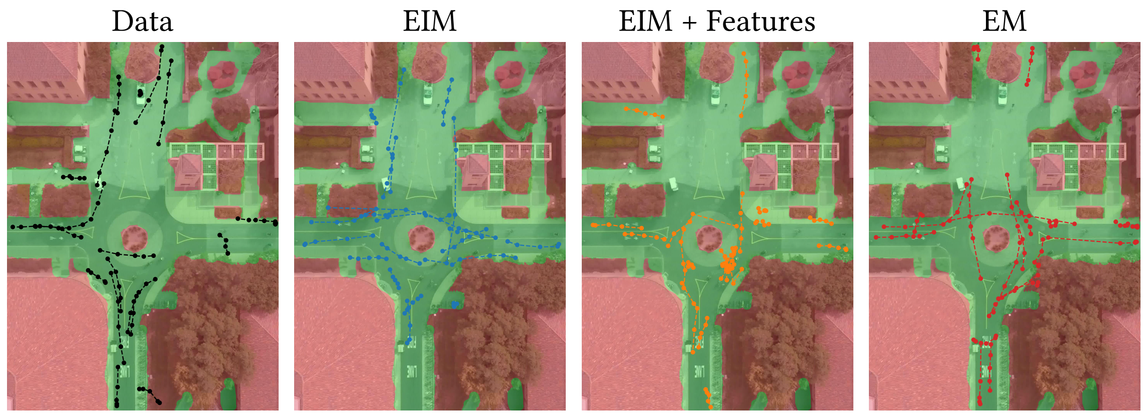


Figure 3.6.: Samples from the Dataset, EIM, EIM with features and EM, plotted over the reference image from the Stanford Drone Dataset and the generated mask. In the mask green corresponds to valid regions and red to invalid regions. EIM with the additional feature input generates samples that stay within the 'road mask' and are therefore considered to be more realistic.

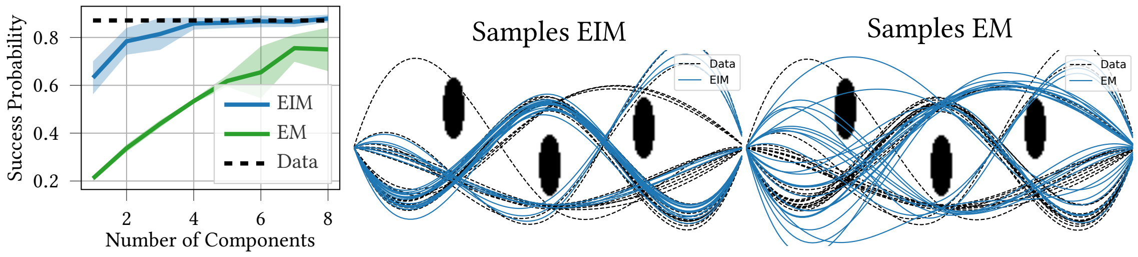


Figure 3.7.: Results on the obstacle avoidance task. Left: Even for a small number of components, EIM has a rather high probability of success, i.e., placing a trajectory that does not hit any obstacle. Even with a sufficient number of components, i.e., eight, EM fails to achieve good results. Middle and Right: Samples of a mixture with 4 components, learned by EIM and EM respectively. EM clearly averages over multiple modes in the data distribution.

the number of components as the test-set likelihood degrades. EIM, on the other hand, produced better models with an increasing number of components. Additionally, we generated a mask indicating whether a given point is on the road or not and evaluated how realistic the learned models are by measuring the amount of samples violating the mask, i.e., predicting road users outside of the road. We also evaluate a version of EIM where we provide additional features indicating if the mask is violated. EIM achieves a much better value on this mask for the NGS dataset, while we needed the additional mask features for the discriminator on the SDD dataset to outperform EM. Both experiments show that EIM can learn highly multimodal density estimates that produce more realistic samples than EM. They further show that the models learned by EIM can be refined by additional prior knowledge provided as feature vectors. Figure 3.6 shows qualitative samples for the different approaches on the Stanford Drone Dataset.

3.5.4. Obstacle Avoidance

We evaluate the conditional version of EIM on an artificial obstacle avoidance task. The context contains the location of three obstacles within an image. The gating, as well as the components, are given by deep neural networks. Details about the network architectures can be found in the Appendix. The data consists of trajectories going from the left to the right of the image. The trajectories are defined by setting 3 via-points such that no obstacle is hit. To generate the data we sample via-points over and under the obstacles with a probability proportional to the distance between the obstacle and the image border. Hence, for three obstacles, there are $2^3 = 8$ different modes in the data. Note that, like in most real-world scenarios, the expert data is not perfect, and about 13% of the trajectories in the dataset hit an obstacle. We fit models with various numbers of components to this data using EIM and EM and compare their performance on generating trajectories that achieve the goal. Results are shown in Figure 3.7 together with a visualization of the task and samples produced by EIM and EM. EIM was able to identify most modes for the different given inputs and did not suffer from any averaging effect. In contrast, EM does not find all modes. As a consequence, some components of the mixture model had to average over multiple modes, resulting in poor quality trajectories.

3.6. Conclusion

We introduced Expected Information Maximization (EIM), a novel approach for computing the I-Projection between general latent variable models and a target distribution, solely based on samples of the latter. General upper bound objectives for marginal and conditional distributions were derived, resulting in an algorithm similar to EM, but tailored for the I-Projection instead of the M-Projection. We introduced efficient methods to optimize these upper bound objectives for mixture models. In our experiments, we demonstrated the benefits of the I-Projection for different behavior modelling tasks. The introduced approach opens various pathways for future research. While we focused on mixture models, the derived upper bounds are not exclusive to those and can be used for arbitrary latent variable models. Another possibility is an online adaptation of the number of used components. Arenz et al. (2018) propose heuristics for such an adaptation in their VIPS approach. Those could easily be adapted to our approach.

Contribution Statement. *I (re)derived the upper bounds for the marginal and derived the bounds for the conditional case. I derived and implemented EIM for Gaussian Mixture Models and Mixtures of Experts, ran all evaluations, and wrote the large majority of the resulting article. OA advised me during the derivation and implementation. He double-checked the derivations, suggested improvements to the implementation, and proposed experiments and ablations. GN suggested using density ratio estimation techniques to extend Variational Inference by Policy Search (Arenz et al., 2018) to density estimation, suggested experiments and wrote small parts of the article.*

3.7. Follow Ups

During my PhD studies, I contributed to several follow-up works that extended Expected Information Maximization (EIM). These projects were led by fellow PhD students or Master's students I supervised, and I list my contributions to the individual works in the contribution statements at the end of the respective sections.

Here, we will briefly discuss three works that build upon ideas from Expectation Information Maximization (EIM). First, Freymuth et al. (2021) introduced Versatile Inverse Reinforcement Learning (VIRL) (Section 3.7.1), an extension of EIM from Imitation Learning to inverse Reinforcement Learning. Next, Freymuth et al. (2022) proposed Versatile Imitation Learning from Geometrically Observed Representations (VIGOR) (Section 3.7.2), a practical Imitation Learning approach building on EIM. It exploits geometric features of planned behavior to train the discriminator and learns multimodal behavior for real-world robots from a few human demonstrations. Freymuth et al. (2021) and Freymuth et al. (2022) use the expression versatile (behavior) synonymously with multimodal (behavior).

Finally, we include Differentiable Trust Region Projection Layers (Otto et al., 2021)(Section 3.7.3). While seemingly unrelated at first glance, the specialized code initially written for EIM proved crucial for this work.

3.7.1. Versatile Inverse Reinforcement Learning via Cumulative Rewards

This section summarizes Versatile Inverse Reinforcement Learning via Cumulative Rewards, *published at the Workshop on Robot Learning: Self-Supervised and Lifelong Learning at NeurIPS 2021 (Niklas Freymuth, Philipp Becker, and Gerhard Neumann, 2021) and draws connections to EIM.*

Inverse Reinforcement Learning (IRL) (Ng and Russell, 2000; Abbeel and Ng, 2004) is a form of Imitation Learning that aims to learn a model of the expert's underlying rewards based on demonstrations. The hope is that this model enables training agents that are more robust than those based on direct imitation, thereby improving performance over the expert and generalizing to novel scenarios. Freymuth et al. (2021) propose Versatile Inverse Reinforcement Learning (VIRL), which builds on EIM to learn reward models from demonstrations of highly versatile behavior by taking the sum of iteratively trained discriminators.

Building on maximum entropy IRL (Ziebart et al., 2008), VIRL assumes the expert's reward is given as $R(\mathbf{x}) = \log p(\mathbf{x}) - c$, where $p(x)$ is the distribution over demonstrations and c is some constant offset. From EIM, we know that we can interactively update a model of the data, $q_t(\mathbf{x})$, by maximizing Equation 3.3, which is optimal for (Abdolmaleki et al., 2015)

$$\log q_{t+1}(\mathbf{x}) = \log q_t(\mathbf{x}) + \phi_t(\mathbf{x}) - c_t. \tag{3.8}$$

Here, $\phi_t(\mathbf{x})$ is a discriminator, approximating the log density ratio between $q_t(\mathbf{x})$ and $p(\mathbf{x})$ and c_t is the log of the normalization constant. If we now assume the algorithm converges after T iterations, i.e., $q_T(\mathbf{x}) = p(x)$, we rewrite the maximum entropy reward model by recursively inserting the solution of Equation 3.8 into itself,

$$
\begin{aligned}
R(\mathbf{x}) &= \log p(\mathbf{x}) - c = \log q_T(\mathbf{x}) - c = \log \exp\left(\log q_{T-1}(\mathbf{x}) - \phi_{T-1}(\mathbf{x}) - c_{T-1}\right) - c \\
&= \log q_0(\mathbf{x}) - \sum_{i=0}^{T-1} (\phi_i(\mathbf{x}) - c_i) - c = \log q_0(\mathbf{x}) - \sum_{i=0}^{T-1} \phi_i(\mathbf{x}) - \tilde{c},
\end{aligned}
$$

where $\tilde{c} = c + \sum_{i=0}^{T-1} c_i$ accumulates all constant offsets and $q_0(\mathbf{x})$ is an arbitrary prior over the demonstrations. In contrast to the parametric model learned by EIM, the cumulative reward function can represent behaviors of arbitrary complexity and is not constrained by the number of model components.

However, VIRL still needs samples from the current $q_t(\mathbf{x})$ to train the discriminators and thus fits a parametric approximation at each iteration using VIPS (Arenz et al., 2018). Crucially, VIRL then corrects these samples using importance weighting, which enables the use of an imperfect approximation to train a discriminator between demonstrations and samples from the current optimal policy.

Furthermore, Freymuth et al. (2021) introduces several practical modifications to stabilize the algorithm, such as enriching the VIPS approximation by a Kernel Density Estimator trained on the data for a more robust sampling. They then evaluate VIRL on two simulated tasks that exhibit highly versatile behavior and demonstrate that VIRL can learn reward functions for these tasks. In particular, the learned reward functions allow training new policies during test time, which cover more modes than the original variational approximation of $q_t^{(x)}$ used for training.

Contribution Statement. *NF derived and implemented VIRL in the course of his Master's Thesis. He made the majority of design choices for the implementation, conducted all experiments and evaluations, and wrote the majority of the paper. I supervised NF during his Master's Thesis, helping with derivations and design choices of the implementation. I also wrote and revised parts of the paper. GN supervised the project, proposed the key idea of using cumulative rewards, and assisted in writing the paper. Both GN and I helped with the experiment design.*

3.7.2. Inferring Versatile Behavior from Demonstrations by Matching Geometric Descriptors

This section summarizes Inferring Versatile Behavior from Demonstrations by Matching Geometric Descriptors*, published in the 6th Conference on Robot Learning, CoRL 2022 (Niklas Freymuth, Nicolas Schreiber, Aleksandar Taranovic, Philipp Becker, and Gerhard Neumann, 2022), and draws connections to EIM.*

Building on EIM, Freymuth et al. (2022) propose Versatile Imitation from Geometrically Observed Representations (VIGOR), a practical Imitation Learning approach applicable

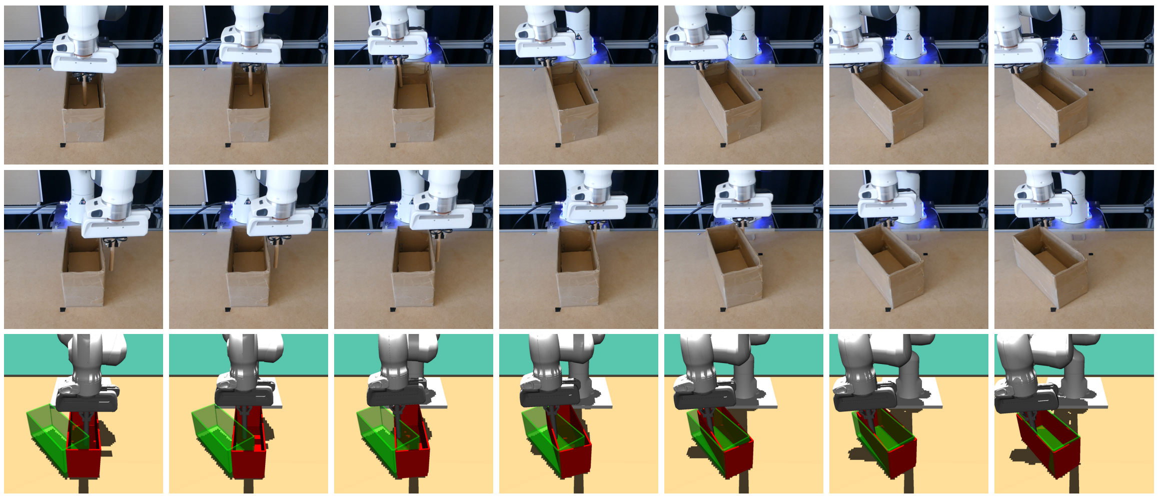


Figure 3.8.: VIGOR (Freymuth et al., 2022) allows Imitation Learning of multimodal behavior on real robots, as demonstrated on the shown task. Here, a Franka Emika Panda robot must push a box into a specified goal pose. The first two rows show executions of the learned behavior on the real robot, while the last row shows the same setup in simulation, with the target indicated in green. The target is the same for all three rows, yet all three executions push the box differently, showing how VIGOR can capture multiple modes in human demonstrations. Here, the first execution pushes from inside the box while the second two push from different locations on the outside. *Figure taken from Freymuth et al. (2022). Caption rewritten.*

to human demonstrations and real-world robots. VIGOR learns versatile behavior, parameterized by mixture models, via a non-adversarial distribution matching approach using a discriminator. This setup is very similar to EIM, but VIGOR fully leans onto the idea of using discriminative features, as discussed in Section 3.3, by relying solely on such features. It utilizes intelligently designed features that exploit geometric relations, abstracting away the concrete task configuration by relying on relative information, e.g., by using the distance between a robot's end-effector and a target instead of its absolute value. Using these features allows VIGOR to learn highly generalizable multimodal behavior from small amounts of contexts and demonstrations on realistic and real robot systems.

VIGOR represents the expert behavior using Probabilistic Movement Primitives (ProMPs) (Paraschos et al., 2013). These compact trajectory representations allow modeling complex trajectories with few parameters $\mathbf{w}$, which massively simplifies learning. Given a context $\mathbf{c}$, VIGOR then fits a mixture model to the ProMP parameters $q(\mathbf{w}|\mathbf{c})$ using an approach closely modeled on EIM. The discriminator fully relies on the sequences of geometric features generated from $\mathbf{w}$, $\mathbf{O}(\mathbf{w}) = (\mathbf{o}_t(\mathbf{w}))_{t=1\cdots T}$ along the entire trajectory, while not considering $\mathbf{w}$ or $\mathbf{c}$ directly. Additionally, VIGOR exploits the sequential nature of the features $\mathbf{O}(\mathbf{w})$ by unfolding the discriminator over time. To this end, a 1d-Convolutional Neural Network provides an output $y_t(\mathbf{O}(\mathbf{w}))$ for each time step given a sequence of features $\mathbf{O}(\mathbf{w})$ with $\mathbf{w}$ either taken from the data or sampled from the current mixture model. The density ratio estimator is trained by minimizing

$$-\mathbb{E}_{q(w|c)}\left[\sum_{t=0}^{T}\log\left(\sigma\left(y_t(\mathbf{O}(\mathbf{w}))\right)\right)\right]-\mathbb{E}_{p(w|c)}\left[\sum_{t=0}^{T}\log\left(1-\sigma\left(y_t(\mathbf{O}(\mathbf{w}))\right)\right)\right],$$

where $p(\mathbf{w}|\mathbf{c})$ is the data distribution and σ the sigmoid activation function.

To learn the actual behavior, VIGOR uses the same update steps as EIM for Gaussian Mixture Models (GMMs) (Section 3.3.3) to learn a single GMM for each training context. This formulation allows using the effective closed-form updates available for marginal models and avoids the costly and noisy reparametrization-based updates for conditional models. Additionally, using a Mixture of Experts model parameterized with a neural network can lead to severe overfitting. This issue is particularly noticeable in scenarios with very few contexts and demonstrations, which is the focus of VIGOR. However, VIGOR can still generalize to previously unseen contexts. During training, VIGOR includes additional contexts for which no expert demonstrations are available, and Freymuth et al. (2022) show that the resulting policies generalize well to these contexts. Additionally, once trained, VIGOR can, in principle, learn behavior for new contexts without additional data by reusing the discriminator. In particular, neither case requires additional interaction with the environment, as the geometric features are evaluated based on the planned behavior given by the ProMP. This approach offers a significant advantage compared to related adversarial Imitation Learning methods (Ho and Ermon, 2016), which heavily rely on such environment interaction during imitation.

Freymuth et al. (2022) evaluated VIGOR on a suite of three tasks, a 2D planar reacher, a 7 Degree of Freedom reach task with a Franka Emika Panda robot, and the box pushing task shown in Figure 3.8. They use human demonstrations for all environments and show how VIGOR can learn highly versatile and accurate behavior from as few as 6 contexts and 5 samples per context. These experiments show how combining geometric features, movement primitives, and ideas from EIM can lead to successful Imitation Learning of multimodal behavior that is transferable from simulation into the real world.

Contribution Statement. *NF derived and implemented VIGOR. He made the majority of design choices for the implementation, conducted the majority of evaluations, including data collection, and wrote the majority of the paper. NS assisted with experiments, particularly in running real-world robot experiments and collecting data. AT participated in discussions about the methodology and experiments. I provided guidance on deriving VIGOR, helped identify implementation issues, and assisted with the experimental design and presentation of the results. I wrote or revised parts of the paper. GN supervised the project, assisted in identifying implementation issues, and suggested relevant experiments.*

3.7.3. Differentiable Trust Region Layers for Deep Reinforcement Learning

This section summarizes Differentiable Trust Region Layers for Deep Reinforcement Learning*, published in the 9th International Conference on Learning Representations (ICLR 2021) (Fabian Otto, Philipp Becker, Vien Anh Ngo, Hanna Carolin Maria Ziesche, and Gerhard Neumann, 2021) and draws connections to EIM.*

Trust Region methods are a crucial tool for model-free on-policy Reinforcement Learning (RL) (Peters et al., 2010), as they can help control exploration and increase the robustness of inherently noisy policy gradient methods. However, enforcing such trust regions in deep RL approaches is notoriously difficult, and many popular approaches resort to crude approximations (Schulman et al., 2015a, 2017). While these approaches can work effectively when properly tuned, they often violate the trust region constraints, fail at exploration, and are challenging to implement due to their high susceptibility to seemingly minor implementation details (Engstrom et al., 2019). As a remedy, Otto et al. (2021) propose differentiable trust region projection layers.

Generally, modern trust region policy gradient methods iteratively collect data with the current policy $\pi_t(\mathbf{a}|\mathbf{s})$, which is then used to update to policy using an importance sampling estimator (Schulman et al., 2015a) by optimizing

$$\begin{aligned}\pi_{k+1}(\mathbf{a}|\mathbf{s}) = &\arg\max_{\pi(\mathbf{a}|\mathbf{s})} \quad \mathbb{E}_{p_0(s_0)\prod_{t=0}^{T}(\pi(a_t|s_t)p(s_{t+1}|s_t,a_t))}\left[\frac{\pi(\mathbf{a}_t|\mathbf{s}_t)}{\pi_k(\mathbf{a}_t|\mathbf{s}_t)}A_k(\mathbf{s}_t,\mathbf{a}_t)\right] \\ &\text{s.t.} \quad d\left(\pi(\mathbf{a}|\mathbf{s}),\pi_k(\mathbf{a}|\mathbf{s})\right) \leq \epsilon \quad \forall \mathbf{s},\end{aligned}$$

where $A_t(\mathbf{s},\mathbf{a})$ denotes the advantage function corresponding to $\pi_t(\mathbf{a}|\mathbf{s})$ and d is some distance measure between probability distributions used to define the trust region. The differentiable trust region projection layers can be added on top of any Gaussian policy $\pi(\mathbf{a}|\mathbf{s}) = \mathcal{N}(\boldsymbol{\mu}(\mathbf{s}),\Sigma(\mathbf{s}))$, parameterized by a neural network. Given $\pi(\mathbf{a}|\mathbf{s})$ and the old policy $\pi_t(\mathbf{a}|\mathbf{s})$ they then project the parameters back inside the trust region, by solving

$$\arg\min_{\tilde{\pi}(\mathbf{a}|\mathbf{s})} \quad d\left(\tilde{\pi}(\mathbf{a}|\mathbf{s}),\pi(\mathbf{a}|\mathbf{s})\right) \quad \text{s.t.} \quad d\left(\tilde{\pi}(\mathbf{a}|\mathbf{s}),\pi_k(\mathbf{a}|\mathbf{s})\right) \leq \epsilon \tag{3.9}$$

for each state. For their practical algorithm, Otto et al. (2021) split the constraint into separate constraints for the policy's mean and covariance, using different trust region sizes for each. Otto et al. (2021) propose three different choices for d, the Frobenius norm between parameters, the Wasserstein-2 (Villani et al., 2008) distance and the KL divergence. While Equation 3.9 can be solved in closed form for Frobenius norm and Wasserstein-2 distance, numerical optimization is required for the KL.

This numerical optimization is where we draw the connection to EIM, as the optimization problem in Equation 3.9 and EIM's trust region-based update for the GMM components are closely related. Both are instances of the general optimization problem considered by Abdolmaleki et al. (2015), which provides a closed-form update for the primal and a numerical solution for the dual optimization problem (Appendix A.1.2). Additionally, for both settings, we need a highly efficient and parallelized implementation as the numerical optimization must be repeatedly solved for a large number of components or states in a batch, respectively. We created such an implementation in C++ during the development of EIM and modified it for the differentiable trust region projection layers by tailoring it to the projection optimization problem and adding gradients by differentiating the KKT conditions following Amos and Kolter (2017).

Among the three measures considered by Otto et al. (2021), the KL projection yielded the best results. It, and our implementation, was adopted by several follow-up works (Otto et al., 2023; Celik et al., 2024; Otto et al., 2025; Hoang et al., 2025).

The mentioned C++ implementation is available at `https://github.com/ALRhub/ITPAL`

Contribution Statement. *FO led the project, did the majority of derivations, and practically realized the trust region projection layers. He implemented the large majority of the code, ran and evaluated the experiments, and wrote the majority of the paper. I wrote the original C++ code for EIM and adapted it for this work. I performed the corresponding derivations, including the calculation of the corresponding gradients. Additionally, I conducted preliminary studies regarding the viability of KL-based trust region projections, VAN proved Theorem 1 in the paper. HZ proposed the Wasserstein-based projection approach. GN proposed the original idea of differentiable trust region projection layers and supervised the project. VAN, HZ, GN, and I assisted with derivations and identifying issues with the implementation.*

4. Recurrent Kalman Networks: Factorized Inference in High-Dimensional Deep Feature Spaces

This chapter establishes a crucial architectural foundation for addressing the second research question, namely the Recurrent Kalman Network (RKN). The RKN combines the strengths of recurrent neural networks (Elman, 1990; Hochreiter and Schmidhuber, 1997) with the probabilistic framework of Kalman filters (Kalman, 1960). By embedding a Linear Gaussian State Space Model (LGSSM) (Section 2.1.3.1) into a recurrent neural network, we obtain a model that can learn to represent complex temporal dependencies in high-dimensional observations while still allowing closed-form inference of belief states in a latent space. To utilize the relatively high-dimensional latent spaces required for modeling complex observations, the RKN relies on a factorized representation, which enables it to remain computationally efficient and scalable. Due to its probabilistic nature and exact inference capabilities, the RKN can model uncertainty in a principled manner and applies to various tasks, including state estimation, filtering, and prediction. This approach is in stark difference to many other advancements in general sequence-to-sequence models, which focus on discrete data and do not holistically consider uncertainty (Hochreiter and Schmidhuber, 1997; Vaswani et al., 2017).

While we focused on state estimation and filtering tasks in the original publication, subsequent work extended the RKN for modeling robotic dynamics, non-stationary systems, and switching dynamics for multimodal behavior. We will briefly summarize these works in Section 4.5. More crucially, in the context of this thesis, the RKN lays the foundation of the subsequently presented work, and its architecture heavily influenced the design of the models presented in Chapter 5 and Chapter 6.

The following was published as Recurrent Kalman Networks: Factorized Inference in High-Dimensional Deep Feature Spaces *(Philipp Becker, Harit Pandya, Gregor Gebhardt, Cheng Zhao, C James Taylor, and Gerhard Neumann, 2019) in the Proceedings of the 36th International Conference on Machine Learning (ICML 2019) Reprinted with permission of the authors. Wording, notation, and formulations were revised in several places. The contents of the original publication were restructured. Section 4.5 briefly summarizes several follow-up papers I contributed to during my PhD studies.*

The work was conducted and published before I started my PhD studies.

4.1. Introduction

State-estimation in unstructured environments is a very challenging task as observations or measurements of the environment are often high-dimensional and only provide partial information about the state. Images are a good example: Even for low resolution, the number of pixels can quickly exceed tens or hundreds of thousands and it is impossible to obtain any information about the dynamics, such as velocities, from a single image. Additionally, the observations may be noisy or may not contain useful information for the task at hand. Such noise can, for example, be introduced by poor illumination or motion blur and occlusions can prevent us from observing some or all relevant aspects of the scene. In addition to state estimation, it is also often desirable to predict future states or observations, for example, in order to assess the consequences of future actions. To this end, an initial estimate of the current state is necessary which again has to be inferred from observations. In such environments, we typically also have to deal with high uncertainties in the state estimates. Being able to model this uncertainty is crucial in many decision making scenarios, e.g., if we need to decide to perform an action now or wait until more information about the scene is available.

Deep learning models have been very successful for time-series modelling in unstructured environments. Classical models such as LSTMs (Hochreiter and Schmidhuber, 1997) or GRUs (Cho et al., 2014) perform well but fail to capture the uncertainty of the state estimate. Recent probabilistic deep learning approaches have used the Kalman filter (KF) as a tool to integrate uncertainty estimates into deep time-series modelling (Haarnoja et al., 2016; Watter et al., 2015; Archer et al., 2015; Fraccaro et al., 2017; Krishnan et al., 2017). These approaches use the KF to perform inference in a low-dimensional (latent) state space that is typically defined by a deep encoder. However, using KF in such a state space comes with two main limitations. In order to be usable for non-linear dynamics, we have to introduce approximations such as the extended KF (Haarnoja et al., 2016) and Variational Inference methods (Krishnan et al., 2017; Fraccaro et al., 2017). Moreover, the KF equations require computationally expensive matrix inversions that are hard to scale to high dimensional latent spaces for more complex systems and computationally demanding to fully backpropagate in an end-to-end manner. Most of these methods are implemented as (variational) auto-encoders and are therefore also limited to predicting future observations or imputing missing observations and can not be directly be applied to state estimation.

We introduce the Recurrent Kalman Network, an end-to-end learning approach for Kalman filtering and prediction. While Kalman filtering in the original state space requires approximations due to the non-linear models, the RKN uses a learned high-dimensional latent state representation that allows for efficient inference using locally linear transition models and a factorized belief state representation. Exploiting this representation allows us to avoid the expensive and numerically problematic matrix inversions involved in the KF equations.

Conceptually, this idea is related to kernel methods which use high-dimensional feature spaces to approximate nonlinear functions with linear models (Gebhardt et al., 2017).

However, in difference to kernel feature spaces, our feature space is given by a deep encoder and learned in an end-to-end manner.

The RKN can be used for any time-series data set for which LSTMs and GRUs are currently the state of the art. In contrast to those, the RKN uses an explicit representation of uncertainty which governs the gating between keeping the current information in memory or updating it with the current observation. While the RKN shows a slightly improved performance in terms of state estimation errors, both LSTMs and GRUs struggle with estimating the uncertainty of the prediction while the RKN can provide accurate uncertainty estimates. In relation to existing KF-based approaches, our approach can be used for state estimation as well as for generative tasks such as image imputation. We also show that we outperform state of the art methods on a complex image imputation task.

4.1.1. Related Work

Using encoders for time-series modelling of high-dimensional data such as images is a common approach. Such encoders can also be easily integrated with well known deep time-series models such as LSTMs (Hochreiter and Schmidhuber, 1997) or GRUs (Cho et al., 2014). These models are very effective but do not provide good uncertainty estimates as shown in our experiments.

Pixel to Torques (P2T) (Wahlström et al., 2015) employs an autoencoder to obtain low dimensional latent representations from images together with a transition model. They subsequently use the models to perform control in the latent space. Embed to Control (E2C) (Watter et al., 2015) can be seen as an extension of the previous approach with the difference that a Variational Autoencoder (Kingma and Welling, 2013) is used. However, both of these approaches are not recurrent and rely on observations which allow inferring the whole state from a single observation. They can therefore not deal with noisy or missing data.

The BackpropKF (Haarnoja et al., 2016) applies a CNN to estimate the observable parts of the true state given the observation. Similar to our approach, this CNN additionally outputs a covariance matrix indicating the model's certainty about the estimate and allows the subsequent use of an (extended) Kalman filter with known transition model. In contrast, we let our model chose the feature space that is used for the inference such that locally linear models can be learned and the KF computations can be simplified due to our factorization assumptions.

Another family of approaches interprets encoder-decoder models as latent variable models that can be optimized efficiently by Variational Inference. They derive a corresponding lower bound and optimize it using the stochastic gradient variational Bayes approach (Kingma and Welling, 2013). Black Box Variational Inference (BB-VI) (Archer et al., 2015) proposes a structured Gaussian variational approximation of the posterior, which simplifies the inference step at the cost of maintaining a tri-diagonal covariance matrix of the full state. To circumvent this issue, Structured Inference Networks (SIN) (Krishnan et al., 2017) employ a flexible recurrent neural network to approximate the dynamic state update. Deep

	scalable	state estimation	uncertainty	noise	direct optimization
LSTM	✓	✓	×/✓	✓	✓
GRU	✓	✓	×/✓	✓	✓
P2T	✓	✓	×/✓	×	✓
E2C	✓	×	✓	×	×
BB-VI	×	×	✓	✓	×
SIN	✓	×	✓	✓	×
KVAE	×	×	✓	✓	×
DVBF	✓	×	✓	✓	×
VSMC	✓	×	✓	✓	×
DSA	✓	×	✓	×	×
DSSM	×	✓ (1D)	✓	×	✓
PRSSM	×	✓	✓	✓	×
RKN	✓	✓	✓	✓	✓

Table 4.1.: Qualitative comparison of our approach to recent related work.

Variational Bayes Filters (DVBF) (Karl et al., 2016) integrate general Bayes filters into deep feature spaces while the Kalman Variational Autoencoder (KVAE) (Fraccaro et al., 2017) employs the classical Kalman Filter and allows not only filtering but also smoothing. Variational Sequential Monte Carlo (VSMC) (Naesseth et al., 2018) uses particle filters instead, however, they are only learning the proposal function and are not working in a learned latent space. Disentangled Sequential Autoencoder (DSA) (Yingzhen and Mandt, 2018) explicitly partitions latent variables for separating static and dynamic content from a sequence of observations however, they do not model state space noise. Yet, these models cannot be directly used for state estimation as they are formulated as generative models of the observations without the notion of the real state of the system. Moreover, the use of Variational Inference introduces an additional approximation that might affect the performance of the algorithms.

Recently, Rangapuram et al. (2018) introduced Deep State Space Models for Time Series Forecasting (DSSM). They employ a recurrent network to learn the parameters of a time varying linear State Space Model (SSM. The emissions of that SSM are the model's final output, i.e. there is no decoder. While this makes the likelihood of the targets given the predicted SSM parameters tractable it limits them to simple latent spaces. Additionally, their approach is only derived for 1 dimensional output data. Probabilistic Recurrent State-Space Models (PRSSM) (Doerr et al., 2018) use Gaussian processes to capture state uncertainties, however, their approach is not demonstrated for high dimensional observations such as images.

A summary of all the approaches and their basic properties can be seen in Table 4.1. We compare whether the approaches are scalable to high dimensional latent states, whether they can be used for state estimation, whether they can provide uncertainty estimates, whether they can handle noise or missing data, and whether the objective can be optimized directly or via a lower bound. All probabilistic generative models rely on Variational Inference which optimizes a lower bound, which potentially affects the performance of the algorithms. P2T and E2C rely on the Markov assumption and therefore can not deal with noise or need very large window sizes. The approaches introduced in (Archer et al., 2015; Fraccaro et al., 2017; Haarnoja et al., 2016) directly use the Kalman update equations in the latent state, which limits these approaches to rather low dimensional latent states due to the expensive matrix inversions.

Traditional recurrent models such as LSTMs or GRUs can be trained directly by backpropagation through time and therefore typically yield very good performance but are lacking uncertainty estimates. However, as in our experiments, they can be artificially added. Our RKN approach combines the advantages of all methods above as it can be learned by direct optimization without the use of a lower bound and it provides a principled way of representing uncertainty inside the neural network.

4.2. Factorized Inference in Deep Feature Spaces

Lifting the original input space to a high-dimensional feature space where linear operations are feasible is a common approach in machine learning, e.g., in kernel regression and SVMs. The Recurrent Kalman Network (RKN) transfers this idea to state estimation and filtering, i.e., we learn a high dimensional deep feature space that allows for efficient inference using the Kalman update equations even for complex systems with high dimensional observations. To achieve this, we assume that the belief state representation can be factorized into independent Gaussian distributions as described in the following sections. Figure 4.1 shows an overview of the proposed architecture.

4.2.1. Latent Observation and State Spaces

The RKN encoder learns a mapping to a high-dimensional latent observation space $\mathcal{W} = \mathbb{R}^m$. The encoder also outputs a vector of uncertainty estimates $\boldsymbol{\sigma}_t^{\mathrm{obs}}$, one for each entry of the latent observation $\mathbf{w}_t$. The latent state space $\mathcal{Z} = \mathbb{R}^n$ of the RKN is related to the observation space $\mathcal{W}$ by the linear latent observation model $\mathbf{H} = \begin{bmatrix} \mathbf{I}_m & \mathbf{0}_{m\times(n-m)} \end{bmatrix}$, i.e., $\mathbf{w} = \mathbf{H}\mathbf{z}$ with $\mathbf{w} \in \mathcal{W}$ and $\mathbf{z} \in \mathcal{Z}$. $\mathbf{I}_m$ denotes the $m \times m$ identity matrix and $\mathbf{0}_{m\times(n-m)}$ denotes a $m \times (n-m)$ matrix filled with zeros.

The idea behind this choice is to split the latent state vector $\mathbf{z}_t$ into two parts, a vector $\mathbf{p}_t$ for holding information that can directly be extracted from the observations and a vector $\mathbf{m}_t$ to store information inferred over time, e.g., velocities. We refer to the former as the observation or upper part and the latter as the memory or lower part of the latent state. For an ordinary dynamical system and images as observations the former may correspond to positions while the latter corresponds to velocities. Clearly, this choice only makes sense for $m \leq n$ and in this work we assume $n = 2m$, i.e., for each observation unit p_i, we also represent a memory unit m_i that stores its velocity information.

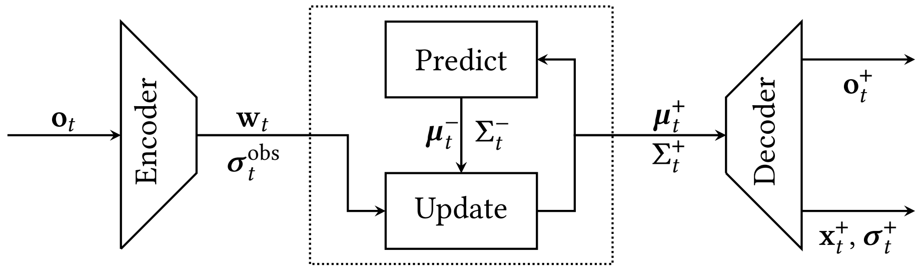


Figure 4.1.: The Recurrent Kalman Network. An encoder network extracts latent features $\mathbf{w}_t$ from the current observation $\mathbf{o}_t$. Additionally, it emits an estimate of the uncertainty in the features via the variance σ_t^{obs}. The transition model $\mathbf{A}_t$ is used to predict the current latent prior $(\boldsymbol{\mu}_t^-, \Sigma_t^-)$ using the last posterior $(\boldsymbol{\mu}_{t-1}^+, \Sigma_{t-1}^+)$ and subsequently update the prior using the latent observation $(\mathbf{w}_t, \sigma_t^{\text{obs}})$. As we use a factorized representation of Σ_t, the Kalman update simplifies to scalar operations. The latent state consists of the observable units as well as the corresponding memory units. Finally, a decoder produces either $(\mathbf{x}_t^+, \sigma_t^+)$, a low dimensional observation and an element-wise uncertainty estimate, or $\mathbf{o}_t^+$, a noise free image.

4.2.2. The Transition Model

To obtain a locally linear transition model we learn K constant transition matrices $\mathbf{A}^{(k)}$ and combine them using coefficients $\alpha^{(k)}(\boldsymbol{\mu}_t^+)$ depending on the current posterior mean $\boldsymbol{\mu}_t^+$, i.e.,

$$\mathbf{A}_t = \sum_{k=1}^{K} \alpha^{(k)}(\boldsymbol{\mu}_t^+)\mathbf{A}^{(k)}.$$

A small neural network with softmax output is used to learn $\alpha^{(k)}$. Similar approaches are used in (Fraccaro et al., 2017; Karl et al., 2016).

Using a dense transition matrix in high-dimensional latent spaces is not feasible as it contains too many parameters and causes numerical instabilities and overfitting, as preliminary experiments showed. Therefore, we design each $\mathbf{A}^{(k)}$ to consist of four band matrices

$$\mathbf{A}^{(k)} = \begin{bmatrix} \mathbf{B}_{11}^{(k)} & \mathbf{B}_{12}^{(k)} \\ \mathbf{B}_{21}^{(k)} & \mathbf{B}_{22}^{(k)} \end{bmatrix}$$

with bandwidth b. Here, $\mathbf{B}_{11}^{(k)}, \mathbf{B}_{12}^{(k)}, \mathbf{B}_{21}^{(k)}, \mathbf{B}_{22}^{(k)} \in \mathbb{R}^{m \times m}$ since we assume $n = 2m$. This choice reduces the number of parameters while not affecting performance since the network is free to choose the state representation.

We assume the covariance of the transition noise to be diagonal and denote the vector containing the diagonal values by $\boldsymbol{\sigma}^{\text{dyn}}$. The noise is learned and independent of the state. Moreover, it is crucial to correctly initialize the transition matrix. Initially, the transition model should focus on copying the encoder output so that the encoder can learn how to extract good features if observations are available and useful. Additionally, it is crucial that $\mathbf{A}$ does not yield an instable system. We choose $\mathbf{B}_{11}^{(k)} = \mathbf{B}_{22}^{(k)} = \mathbf{I}$ and $\mathbf{B}_{12}^{(k)} = -\mathbf{B}_{21}^{(k)} = 0.2 \cdot \mathbf{I}$.

4.2.3. Factorized Covariance Representation

Since the RKN learns high-dimensional representations, we can not work with the full state covariance matrices Σ_t^+ and Σ_t^-. We can also not fully factorize the state covariances by diagonal matrices as this neglects the correlation between the memory and the observation parts. As the memory part is excluded from the observation model $\mathbf{H}$, the Kalman update step would not change the memory units nor their uncertainty if we would only use a diagonal covariance matrix Σ_t^+. Hence, for each observation unit p_i, we compute the covariance with its corresponding memory unit m_i. All the other covariances are neglected. This might be a crude approximation for many systems, however, as our network is free to choose its own state representation it can find a representation where such a factorization works well in practice. Thus, we use matrices of the form

$$\Sigma_t = \begin{bmatrix} \Sigma_t^{\mathrm{u}} & \Sigma_t^{\mathrm{s}} \\ \Sigma_t^{\mathrm{s}} & \Sigma_t^{\mathrm{l}} \end{bmatrix}$$

where each of $\Sigma_t^{\mathrm{u}}, \Sigma_t^{\mathrm{s}}, \Sigma_t^{\mathrm{l}} \in \mathbb{R}^{m \times m}$ is a diagonal matrix. Again, we denote the vectors containing the diagonal values by $\boldsymbol{\sigma}_t^{\mathrm{u}}, \boldsymbol{\sigma}_t^{\mathrm{l}}$ and $\boldsymbol{\sigma}_t^{\mathrm{s}}$.

4.2.4. Factorized Inference in the Latent Space

Inference in the latent state space can now be implemented, similar to a standard KF, using a prediction and an observation update. Following the factorized covariance representation introduced above, we split the means of prior and posterior beliefs for these derivation. The observation and memory parts are denoted by $\boldsymbol{\mu}_t^{\mathrm{u}}$ and $\boldsymbol{\mu}_t^{\mathrm{l}}$, respectively.

Prediction Update. Equivalently to the classical Kalman Filter, the next prior $\left(\boldsymbol{\mu}_{t+1}^-, \Sigma_{t+1}^-\right)$ is obtained from the current posterior $\left(\boldsymbol{\mu}_t^+, \Sigma_t^+\right)$ by

$$\boldsymbol{\mu}_{t+1}^- = \mathbf{A}_t \boldsymbol{\mu}_t^+ \quad \text{and} \quad \Sigma_{t+1}^- = \mathbf{A}_t \Sigma_t^+ \mathbf{A}_t^T + \mathbf{I} \cdot \boldsymbol{\sigma}^{\mathrm{dyn}}.$$

However, the special structure of the covariances enables us to significantly simplify the equation for the covariance. While being straight forward, the full derivations are rather lengthy, thus, we refer to the supplementary material where the equations are given in Equation B.1, Equation B.2 and Equation B.3.

Observation Update. Next, the prior is updated using the latent observation $(\mathbf{w}_t, \boldsymbol{\sigma}_t^{\mathrm{obs}})$. Similar to the state, we split the Kalman gain matrix $\mathbf{Q}_t$ into an upper $\mathbf{Q}_t^{\mathrm{u}}$ and a lower part $\mathbf{Q}_t^{\mathrm{l}}$. Both $\mathbf{Q}_t^{\mathrm{u}}$ and $\mathbf{Q}_t^{\mathrm{l}}$ are squared matrices. Due to the simple latent observation model $\mathbf{H} = \begin{bmatrix} \mathbf{I}_m & \mathbf{0}_{m \times (n-m)} \end{bmatrix}$ and the factorized covariances all off-diagonal entries of $\mathbf{Q}_t^{\mathrm{u}}$ and $\mathbf{Q}_t^{\mathrm{l}}$ are zero and we can work with vectors representing the diagonals, i.e., $\mathbf{q}_t^{\mathrm{u}}$ and $\mathbf{q}_t^{\mathrm{l}}$. Those are obtained by

$$\mathbf{q}_t^{\mathrm{u}} = \boldsymbol{\sigma}_t^{\mathrm{u},-} \oslash \left(\boldsymbol{\sigma}_t^{\mathrm{u},-} + \boldsymbol{\sigma}_t^{\mathrm{obs}}\right) \quad \text{and} \quad \mathbf{q}_t^{\mathrm{l}} = \boldsymbol{\sigma}_t^{\mathrm{s},-} \oslash \left(\boldsymbol{\sigma}_t^{\mathrm{u},-} + \boldsymbol{\sigma}_t^{\mathrm{obs}}\right),$$

where $\oslash$ denotes an elementwise vector division. With this the update equation for the mean simplifies to

$$\boldsymbol{\mu}_t^+ = \boldsymbol{\mu}_t^- + \begin{bmatrix} \mathbf{q}_t^{\mathrm{u}} \\ \mathbf{q}_t^{\mathrm{l}} \end{bmatrix} \odot \begin{bmatrix} \mathbf{w}_t - \boldsymbol{\mu}_t^{\mathrm{u},-} \\ \mathbf{w}_t - \boldsymbol{\mu}_t^{\mathrm{u},-} \end{bmatrix}$$

where $\odot$ denotes the elementwise vector product. The update equations for the individual covariance parts are given by

$$\boldsymbol{\sigma}_t^{\mathrm{u},+} = \left(\mathbf{1}_m - \mathbf{q}_t^{\mathrm{u}}\right) \odot \boldsymbol{\sigma}_t^{\mathrm{u},-}, \ \boldsymbol{\sigma}_t^{\mathrm{s},+} = \left(\mathbf{1}_m - \mathbf{q}_t^{\mathrm{u}}\right) \odot \boldsymbol{\sigma}_t^{\mathrm{s},-}, \text{ and } \boldsymbol{\sigma}_t^{\mathrm{l},+} = \boldsymbol{\sigma}_t^{\mathrm{l},-} - \mathbf{q}_t^{\mathrm{l}} \odot \boldsymbol{\sigma}_t^{\mathrm{s},-},$$

where $\mathbf{1}_m$ denotes the m dimensional vector consisting of ones. Again, we refer to the supplementary material for a more detailed derivations.

Besides avoiding the matrix inversion in the Kalman gain computation, the factorization of the covariance matrices reduces the total amount of numbers to store per matrix from n^2 to $3m$. Additionally, working with $\boldsymbol{\sigma}^{\mathrm{s}}$ makes it trivial to ensure that the symmetry and positive definiteness of the state covariance are not affected by numerical issues.

Initialization. We initialize $\boldsymbol{\mu}_0^+$ with an all zeros vector and Σ_0^+ with $10 \cdot \mathbf{I}$. In practice, it is beneficial to normalize $\mathbf{w}_t$ since the statistics of noisy and noise free images differ considerably.

4.2.5. Loss and Training

We consider two different potential output distributions, Gaussian distributions for estimating low dimensional observations and Bernoulli distributions for predicting high dimensional observations such as images. Both distributions are computed by decoders that use the current latent state estimate $\boldsymbol{\mu}_t^+$ as well its uncertainty estimates $\boldsymbol{\sigma}_t^{\mathrm{u},+}, \boldsymbol{\sigma}_t^{\mathrm{s},+}, \boldsymbol{\sigma}_t^{\mathrm{l},+}$.

Inferring States. Let $\mathbf{x}_{\leq T}$ be the ground truth sequence with dimension D_x, the Gaussian log-likelihood for a single sequence is then computed as

$$\mathcal{L}\left(\mathbf{x}_{\leq T}\right) = \frac{1}{T} \sum_{t=1}^{T} \log \mathcal{N}\left(\mathbf{x}_t \middle| \operatorname{dec}_\mu(\boldsymbol{\mu}_t^+), \operatorname{dec}_\Sigma(\boldsymbol{\sigma}_t^{\mathrm{u},+}, \boldsymbol{\sigma}_t^{\mathrm{s},+}, \boldsymbol{\sigma}_t^{\mathrm{l},+})\right),$$

where $\operatorname{dec}_\mu(\cdot)$ and $\operatorname{dec}_\Sigma(\cdot)$ denote the parts of the decoder that are responsible for decoding the latent mean and latent variance respectively.

Inferring Images. Let $\mathbf{o}_{\leq T}$ be images with D_o pixels. The Bernoulli log-likelihood for a single sequence is then given by

$$\mathcal{L}\left(\mathbf{o}_{\leq T}\right) = \frac{1}{T}\sum_{t=1}^{T}\sum_{i=0}^{D_\text{o}} o_t^{(i)} \log\left(\text{dec}_{o,i}\left(\boldsymbol{\mu}_t^+\right)\right) + \left(1 - o_t^{(i)}\right)\log\left(1 - \text{dec}_{o,i}\left(\boldsymbol{\mu}_t^+\right)\right),$$

where $o_t^{(i)}$ is the ith pixel of the tth image. The pixels are in this case represented by grayscale values in the range of $[0;1]$. The ith dimension of the decoder is denoted by $\text{dec}_{o,i}\left(\boldsymbol{\mu}_t^+\right)$, where we use a sigmoid transfer function as output units for the decoder.

Gradients are computed using (truncated) backpropagation through time (BPTT) (Werbos, 1990) and clipped. We optimize the objective using the Adam (Kingma and Ba, 2014) stochastic gradient descent optimizer with default parameters.

4.2.6. The Recurrent Kalman Network

The prediction and observation updates result in a new type of recurrent neural network, that we call Recurrent Kalman Network, which allows working in high dimensional state spaces while keeping numerical stability, computational efficiency and (relatively) low memory consumption.

Similar to the input gate in LSTMs (Hochreiter and Schmidhuber, 1997) and GRUs (Cho et al., 2014) the Kalman gain can be seen as a gate controlling how much the current observation influences the state. However, this gating explicitly depends on the uncertainty estimates of the latent state and observation and is computed in a principled manner. While the sparse transition models and factorization assumptions may seem restrictive, they allow stable and efficient computations in high dimensional spaces. Since the high dimensional representation is learned jointly with the dynamics, this can yield very powerful models, as shown in our experiments.

In comparison to LSTMs and GRUs the number of parameters is considerably smaller. For a fixed bandwidth b and number of basis matrices k it scales linearly in the state size for the RKN while it scales quadratically for LSTMs and GRUs.

Moreover, the RKN provides a principled method to deal with absent inputs by just omitting the update step and setting the posterior to the prior.

4.3. Evaluation and Experiments

A full listing of hyperparameters and data set specifications can be found in the supplementary material. We compare to LSTM and GRU baselines for which we replaced the RKN transition layer with generic LSTM and GRU layers. Those were given the encoder output as inputs and have an internal state size of $2n$. The internal state was split into two equally large parts, the first part was used to compute the mean and the second to compute the

Model	Log Likelihood
RKN $(m = 15, b = 3, K = 15)$	6.182 ± 0.155
RKN $m = b = 15, K = 3)$	6.248 ± 0.1715
RKN $(m = 15, b = 3, K = 15,$ fc $)$	6.161 ± 0.23
RKN $(m = b = 15, K = 15,$ fc $)$	6.197 ± 0.249
LSTM $m = 50$	5.773 ± 0.231
LSTM $m = 6$	6.019 ± 0.122
GRU $m = 50$	5.649 ± 0.197
GRU $m = 8$	6.051 ± 0.145

Table 4.2.: Our approach outperforms the generic LSTM and GRU baselines. The GRU with $m = 8$ and the LSTM with $m = 6$ were designed to have roughly the same amount of parameters as the RKN with $b = 3$. In the case where $m = b$ the RKN uses a full transition matrix. *fc* stands for full covariance matrix, i.e., we do not use factorization of the belief state.

variance. We additionally executed most of the following experiments using the root mean squared error to illustrate that our approach is also competitive in prediction accuracy. The RMSE results can be found in the appendix.

4.3.1. Pendulum

We evaluated the RKN on a simple simulated pendulum with images of size 24×24 pixels as observations. Gaussian transition noise with standard deviation of $\sigma = 0.1$ was added to the angular velocity after each step. In the first experiment, we evaluated filtering in the presence of high observation noise. We compare against LSTMs and GRUs as these are the only methods that can also perform state estimation. The amount of noise varies between no noise at all and the whole image consisting of pure noise. Furthermore, the noise is correlated over time, i.e., the model may observe pure noise for several consecutive time steps. Details about the noise sampling process can be found in the appendix. We represent the joint angle θ_t as a two dimensional vector $\mathbf{s}_t = \boldsymbol{\theta}_t = (\sin(\theta_t), \cos(\theta_t))^T$ to avoid discontinuities. We compared different instances of our model to evaluate the effects of assuming sparse transition models and factorized state covariances. The results are given in Table 4.2. The results show that our assumptions do not affect the performance, while we need to learn fewer parameters and can use much more efficient computations using the factorized representation. Note that for the more complex experiments with a higher dimensional latent state space, we were not able to experiment with full covariance matrices due to lack of memory and massive computation times. Moreover, the RKN outperforms LSTM and GRU in all settings.

In a second experiment, we evaluated the image prediction performance and compare against existing Variational Inference approaches. We randomly removed half of the images from the sequences and tasked the models with imputing those missing frames, i.e., we train the models to predict images instead of the position. We compared our approach to the Kalman Variational Autoencoder (KVAE) (Fraccaro et al., 2017), Embed to Control

Model	Log Likelihood
RKN (informed)	-12.782 ± 0.0160
RKN (uninformed)	-12.788 ± 0.0142
KVAE (informed, filter)	-14.383 ± 0.229
KVAE (informed, smooth)	-13.337 ± 0.236
KVAE (uninformed, filter)	-46.320 ± 6.488
KVAE (uninformed, smooth)	-38.170 ± 5.399
E2C (informed)	-95.539 ± 1.754
SIN (informed)	-101.268 ± 0.567

Table 4.3.: Comparison on the image imputation task. The informed models were given a mask of booleans indicating which images are valid and which not. The uninformed models were given a black image whenever the image was not valid. E2C and SIN only work in the informed case. Since the KVAE is capable of smoothing in addition to filtering, we evaluated both. Our approach outperforms all models. Only the informed KVAE yields comparable, but still slightly worse results while E2C and SIN fail to capture the dynamics. The uninformed KVAE fails at identifying the invalid images.

(E2C) (Watter et al., 2015) and Structured Inference Networks (SIN) (Krishnan et al., 2017). The results can be found in Table 4.3. Again, our RKN outperforms all other models. This is surprising as the Variational Inference models use much more complex inference methods and in some cases even more information such as in the KVAE smoothing case. Sample sequences can be found in the supplementary material. All hyperparameters are the same as for the previous experiment.

4.3.2. Multiple Pendulums

Dealing with uncertainty in a principled manner becomes even more important if the observation noise affects different parts of the observation in different ways. In such a scenario the model has to infer which parts are currently observable and which parts are not. To obtain a simple experiment with that property, we repeated the pendulum experiments with three colored pendulums in one image. The image was split into four equally sized square parts and the noise was generated individually for each part such that some pendulums may be occluded while others are visible. Exemplary images are shown in the supplementary material. A comparison to LSTM and GRU baselines can be found in Figure 4.2 and Figure 4.2 as well as an exemplary trajectory in Figure 4.3. The RKN again clearly outperforms the competing methods. We also computed the quality of the uncertainty prediction by showing the histograms of the normalized prediction errors. While this histogram has clearly a Gaussian shape for the RKN, it looks like a less regular distribution for the LSTMs.

Model	Log Likelihood
RKN $m = 45$ $b = 3, k = 15$	11.51 ± 1.703
LSTM $m = 50$	7.5224 ± 1.564
LSTM $m = 12$	7.429 ± 1.307
GRU $m = 50$	7.541 ± 1.547
GRU $m = 12$	5.602 ± 1.468

Table 4.4.: Results of the multiple pendulum experiments.

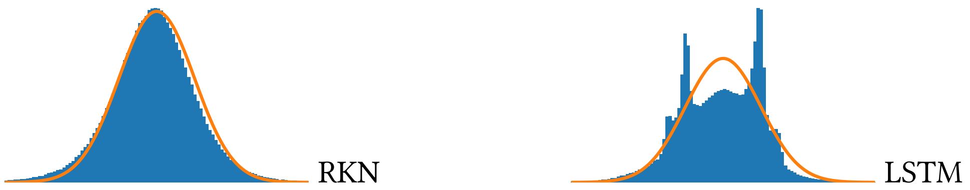


Figure 4.2.: To evaluate the quality of our uncertainty prediction we compute the normalized error $\frac{x^{(j)}-x^{(j),+}}{\sigma^{(j),+}}$ for each entry j of $\mathbf{x}$ for all time steps in all test sequences. This normalized error should follow a Gaussian distribution with unit variance if the prediction is correct. We compare the resulting error histograms with a unit variance Gaussian. The left histogram shows the RKN, the right one the LSTM. The RKN perfectly fits the normal distribution while the LSTM's normalized error distribution has several modes. Again we designed the smaller LSTM and GRU to have roughly the same amount of parameters as the RKN.

4.3.3. Quad Link

We repeated the filtering experiment on a system with much more complicated dynamics, a quad link pendulum on images of size 48×48 pixels. Since the individual links of the quad link may occlude each other different amounts of noise are induced for each link. Two versions of this experiment were evaluated. One without additional noise and one were we added noise generated by the same process used in the pendulum experiments. You can find the results in Table 4.5.

Furthermore, we repeated the imputation experiment with the quad link. We compared only to the informed KVAE, since it was the only model producing competitive results for the pendulum. Our approach achieved -44.470 ± 0.208 (informed) and -44.584 ± 0.236 (uninformed). The KVAE achieved -52.608 ± 0.602 for smoothing and -59.0218 ± 0.580 for filtering . Sample images can be found in the appendix.

4.3.4. KITTI Dataset for Visual Odometry

Next, we evaluate the RKN on the task of learning visual odometry from monocular images on the KITTI visual odometry dataset (Geiger et al., 2012). It consists of 11 labeled image sequences collected by driving a vehicle in an urban environment.

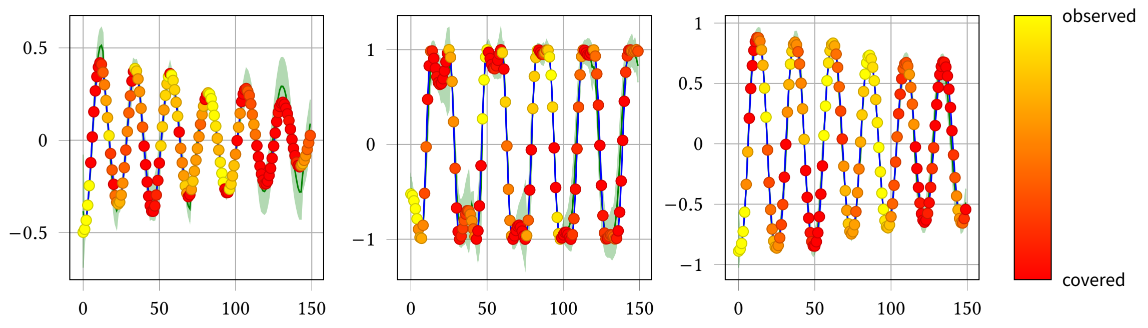


Figure 4.3.: Predicted sine value of the three links with 2 times standard deviation (green). Ground truth displayed in blue. The crosses visualize the current visibility of the link with yellow corresponding to fully visible and red to fully occluded. If there is no observation for a considerable time the predictions become less precise due to transition noise, however, the variance also increases.

Model	without noise Log Likelihood	with noise Log Likelihood
RKN $(m = 100, b = 3, K = 15)$	14.534 ± 0.176	6.259 ± 0.412
LSTM $(m = 50)$	11.960 ± 1.24	5.21 ± 0.305
LSTM $(m = 100)$	7.858 ± 4.680	3.87 ± 0.938
GRU $(m = 50)$	10.346 ± 2.70	4.696 ± 0.699
GRU $(m = 100)$	5.82 ± 2.80	1.2 ± 1.105

Table 4.5.: Comparison of our approach with the LSTM and GRU Baselines on the Quad Link Pendulum. Again the RKN performs significantly better than LSTM and GRU who fail to perform well.

We use the unsupervised approach proposed by Zhou et al. (2017) to extract features from the images. Using these features, we first evaluate a simple forward network, which we refer to as sfmlearner-s. We then augment this architecture with LSTMs, GRUs, and our proposed RKN. Additionally, we compare to DeepVO (Wang et al., 2017), an approach tailored to the problem of visual odometry. In Table 4.6 we show results using the common KITTI evaluation scheme. The results show that the RKN shows better performance in comparison to LSTMs and GRUs, and even performs comparably to the tailored DeepVO approach.

4.3.5. Pneumatic Brook Robot Arm

We consider another real world task, learning the dynamics of a pneumatically actuated joint of a Hydro-Lek HLK-7W robot arm. The data consists of measured joint positions and the input current to the controller of the joint sampled at 100Hz. Pneumatic robots are very hard to model as there is a large amount of hysteresis, requiring recurrent prediction models.

During training, we work with sequences of length 500. For the first 300 time steps those sequences consist of the full observation, i.e. joint position and current. The remaining

Seq.	Deep VO $t_{\text{rel}}(\%)$	Deep VO $r_{\text{rel}}(^\circ)$	sfmlearner-s $t_{\text{rel}}(\%)$	sfmlearner-s $r_{\text{rel}}(^\circ)$	LSTM $t_{\text{rel}}(\%)$	LSTM $r_{\text{rel}}(^\circ)$	GRU $t_{\text{rel}}(\%)$	GRU $r_{\text{rel}}(^\circ)$	RKN $t_{\text{rel}}(\%)$	RKN $r_{\text{rel}}(^\circ)$
3	8.49	6.89	13.21	6.70	8.99	4.55	9.34	3.81	7.83	3.57
4	7.19	6.97	13.56	4.56	11.88	3.44	12.36	2.89	11.61	2.61
5	2.62	3.61	13.06	5.70	8.96	3.43	10.02	3.43	7.29	2.77
6	5.42	5.82	10.87	4.47	9.66	2.80	10.99	3.22	8.08	2.32
7	3.91	4.60	13.47	8.41	9.83	5.48	13.70	6.52	9.38	4.83
10	8.11	8.83	16.69	6.26	13.85	3.49	13.37	3.25	12.71	3.09
mean	5.96	6.12	13.48	6.02	10.53	3.87	11.63	3.85	9.48	3.20

Table 4.6.: Comparison on the KITTI dataset using the common evaluation scheme. The RKN performs better than the non-recurrent sfmlearner-s baseline as well as the recurrent LSTM and GRU baselines. It performs comparably to DeepVO (Wang et al., 2017), an approach tailored to the task of visual odometry.

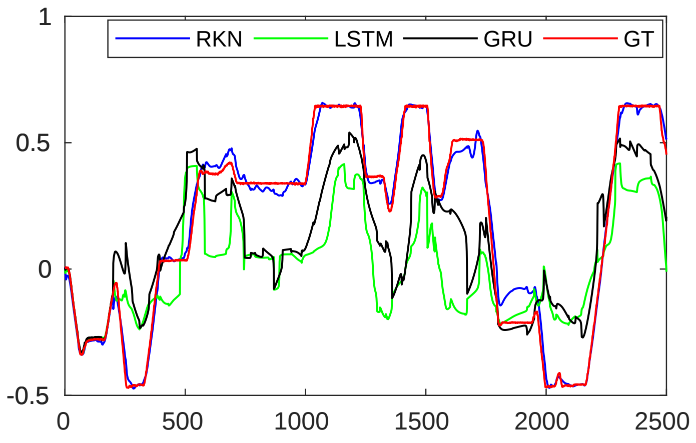


Figure 4.4.: Predicted joint values for the pneumatic joint prediction task. The initial burn-in phase is 200 time steps long, afterwards only the predictions are displayed. GT denotes the ground truth. As can be seen, only our approach manages to give feasible predictions.

200 time steps only the current is given. The models have to impute the missing joint positions in an uninformed fashion, i.e., the absence of a position is only indicated by unrealistically high values.

In order to evaluate the learned dynamics, we used them to predict future joint positions. After an initial burn-in phase, the model was tasked with predicting the joint position 2 seconds into the future, given only the future control inputs. Afterwards, the next observation was given to the model and the prediction process repeated.

Using a 60 dimensional latent space, 32 basis matrices and a bandwidth of 3, the RKN achieved a log likelihood of 4.930. This is considerably better than the LSTM and GRU baselines, which achieved 0.952 and 1.186 respectively. Figure 4.4 shows exemplary predictions of all approaches.

4.4. Conclusion

In this paper, we introduced the Recurrent Kalman Network that jointly learns high-dimensional representations of the system in a latent space with locally linear transition models and factorized covariances. The update equations in the high-dimensional space are based on the update equations of the classical Kalman filter, however, due to the factorization assumptions, they simplify to scalar operations that can be performed much faster and with greater numerical stability. Our model outperforms generic LSTMs and GRUs on various state estimation tasks while providing reasonable uncertainty estimates. Additionally, it outperformed several generative models on an image imputation task and we demonstrated its applicability to real world scenarios. Training is straight forward and can be done in an end-to-end fashion.

In future work, we want to exploit the principled notion of a variance provided by our approach in scenarios where such a notion is beneficial, e.g., Reinforcement Learning. Similar to Fraccaro et al. (2017) we could expand our approach to not just filter but smooth over trajectories in offline post-processing scenarios which could potentially increase the estimation performance significantly. Additionally, different, more complex, factorization assumptions can be investigated in the future.

Contribution Statement. *I devised and derived the factorized inference scheme, implemented the model, devised and ran the large majority of the evaluations, and wrote the large majority of the article. HP conducted the KITTI evaluation, helped with the Pneumatic Arm experiment and wrote the corresponding parts of the article. GG advised me during the project, helped with devising approach and experiments, and guided me in writing the article. CZ helped with the experimental evaluation. JT provided the dataset for the Pneumatic Brook Robot Arm and helped with writing the corresponding part of the article. GN suggested the initial idea of combining Kalman filtering and deep learning, advised during the project, helped with devising the approach and experiments, and guided me in writing the article.*

4.5. Follow Ups

During my PhD studies, I contributed to several follow-up works, extending the Recurrent Kalman Network (RKN). These projects were led by fellow PhD students or Master's students I supervised. I list my contributions to the individual works in the contribution statements at the end of the respective sections.

Here, we will briefly discuss three works that build upon ideas from the Recurrent Kalman Network (RKN). First, Shaj et al. (2020) introduced the Action-Conditional Recurrent Kalman Network (ACRKN), a principled method for incorporating control inputs into the RKN, which was used for learning forward and inverse dynamics in robotics. Next, Shaj et al. (2022) introduced Hidden Parameter Recurrent State Space Models (HiPRSSMs), extending the ACRKN to learn non-stationary dynamics by leveraging additional conditional latent variables to capture the non-stationarity of the system. These architectures, in particular the ACRKN, inspired the VRKN's (Chapter 5) and KalMamba's (Chapter 6) dynamics parameterization. Finally, Nguyen-Quynh et al. (2021) introduced Switching Recurrent Kalman Networks (SRKN). The SRKN builds upon the previously explored theme of modeling multimodal human behavior (Chapter 3), yet it focuses on prediction rather than imitation. Furthermore, it employs a fully variational lower bound objective, which precludes some aspects of the VRKN's training scheme (Chapter 5) without learning an action conditional model or utilizing smoothing inference.

4.5.1. Action-Conditional Recurrent Kalman Networks for Forward and Inverse Dynamics Learning

This section summarizes Action-Conditional Recurrent Kalman Networks for Forward and Inverse Dynamics Learning*, published in the 4th Conference on Robot Learning (CoRL 2020), (Vaisakh Shaj, Philipp Becker, Dieter Buchler, Harit Pandya, Niels van Duijkeren, C James Taylor, Marc Hanheide, and Gerhard Neumann, 2020) and draws connections to the RKN and the works presented in the following chapters.*

The Action-Conditional Recurrent Kalman Network (ac-RKN) (Shaj et al., 2020) extends the RKN to systems with control inputs, taking an important step towards using it for world modeling. While the RKN could, in principle, handle actuated systems by using the control input as an additional observation, this does not align with the State Space Model's assumptions. To make the action conditioning more principled, the ac-RKN builds on the RKN's locally linear dynamics model and introduces the action conditioning in the predict step. The resulting dynamics model is given by

$$\mathcal{N}(\mathbf{z}_{t+1}|\mathbf{A}_t\mathbf{z}_t + b(\mathbf{a}_t), \sigma^{\text{dyn}}\mathbf{I})$$

where b is a function mapping the control input to the latent space. Importantly, this formulation assumes uncertainty-free control inputs based on the argument that the

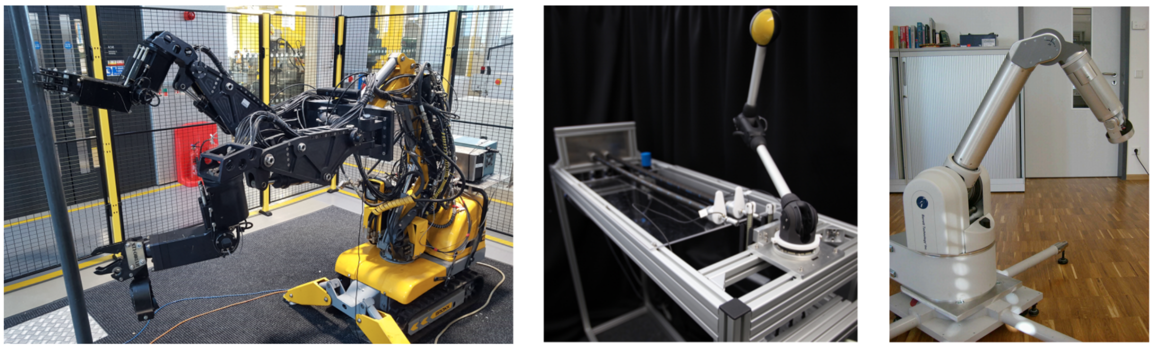


Figure 4.5.: From left to right: The hydraulic BROKK-40 Robot arm, the 4 DoF pneumatically actuated musculoskeletal robot arm, and a cable-driven Barret WAM used to demonstrate the ac-RKN's effectiveness. The ac-RKN combines ideas from the RKN with a more principled action conditioning scheme and a predictive training objective to learn precise forward and inverse dynamics models of the highly challenging dynamics of these robots. *Figure adapted from Shaj et al. (2020). Caption rewritten.*

control input is deterministic. These assumptions result in the following predict step to propagate the latent belief to the next time step

$$\boldsymbol{\mu}_{t+1}^{-} = \mathbf{A}_t \boldsymbol{\mu}_t + b(\mathbf{a}_t) \text{ and } \Sigma_{t+1}^{-} = \mathbf{A}_t \Sigma_t \mathbf{A}_t^{\top} + \sigma^{\text{dyn}} \mathbf{I}.$$

Crucially, this closed-form update does not make any assumptions on the function b, which allows using a nonlinear neural network to model the action conditioning. Shaj et al. (2020) compare this choice against globally and locally linear models and, unsurprisingly, find the nonlinear model to perform best in most scenarios. Finally, Shaj et al. (2020) retain the RKN's factorized representation of the latent state, the parameterization of the transition matrix $\mathbf{A}_t$, and the Kalman filter-based inference scheme. The ac-RKN employs the same direct likelihood-based training objective as the RKN, but it predicts the target observations from the current prior rather than the current posterior. This difference tailors the model more towards prediction instead of state estimation, allowing multi-step prediction as it forces the state representation to remain consistent over time.

Besides these forward dynamics models, Shaj et al. (2020) present ac-RKN models for inverse dynamics learning. They collect real-world data from several robots, including the hydraulic BROKK-40 Robot Arm already considered with the RKN, a four degree of freedom (DoF) pneumatically actuated musculoskeletal robot (PAM) (Büchler et al., 2016), a cable-driven Barret WAM and a Franka Emika Panda. See Figure 4.5 for pictures of some of the robots used in the experiments. Except for the Franka Emika Panda, the nature of these robots and their actuation makes it impossible to describe them using rigid body dynamics and brings more advanced traditional modeling techniques to their limits. The ac-RKN excels at capturing these complex dynamics and outperforms traditional methods, standard recurrent neural networks (Hochreiter and Schmidhuber, 1997; Cho et al., 2014), and the RKN, which employs a more naive approach to action conditioning using the observations.

While shifting the focus back to more general world modeling for embodied agents, the methods introduced in Chapter 5 and Chapter 6 are both heavily influenced by the ac-RKN, in particularly its action conditioning scheme. They take the idea of the ac-RKNs nonlinear action conditioning even further by using a neural network to parameterize all of the dynamics models parameters, including the transition matrix $\mathbf{A}_t$ and the noise covariance Σ_t^{dyn}. In particular, these methods learn a time-dependent transition noise variance taking, among other quantities, the action as input. This parameterization lifts the ac-RKN's assumption of a constant transition noise variance, accounting for the fact that different actions can lead to different noise levels. For example, in many applications, larger actions lead to faster movements and, thus, larger uncertainties in their effects.

On the one hand, the way the ac-RKN learns forward dynamics by reconstructing observations from the prior is similar to the predictive coding approach used in Chapter 7. On the other hand, it differs from the approaches used in Chapter 5 and Chapter 6, as well as the variational approach in Chapter 7, where the KL term naturally arising from the lower bound trains the dynamics model and enforces consistency between the prior and posterior.

Contribution Statement. *VS led the projects and designed the ac-RKN for forward and inverse dynamics learning, including the training objectives. He conducted the experiments and wrote large parts of the paper. I provided the initial RKN implementation and assisted with devising the algorithm, implementing details, conducting experiments, and writing the article. DB provided the data for the musculoskeletal robot, assisted in conducting the corresponding experiments, and offered feedback on the paper. HP and NvD assisted with experimental design and execution as well as writing the paper. JT provided the data for the BROKK-40 robot and provided feedback on the corresponding parts of the paper. MH and GN supervised the project and provided feedback on the paper. GN also contributed to the design of the method and experiments.*

4.5.2. Hidden Parameter Recurrent State Space Models For Changing Dynamics Scenarios

This section summarizes Hidden Parameter Recurrent State Space Models For Changing Dynamics Scenarios, *published in the 10th International Conference on Learning Representations (ICLR 2022) (Vaisakh Shaj, Dieter Büchler, Rohit Sonker, Philipp Becker, and Gerhard Neumann, 2022) and draws connections to the RKN.* Building on the ac-RKN, Shaj et al. (2022) introduced the Hidden Parameter Recurrent State Space Model (HiPRSSM) to enable learning dynamics models under non-stationary conditions. Such non-stationary effects arise in many real-world applications, for example, due to wear and tear or changing environmental conditions such as temperature, and are poorly captured by state space approaches not explicitly designed for this purpose. To account for these effects, the HiPRSSM introduces a hidden task variable $\mathbf{l}$, i.e., an additional latent variable that is assumed to be constant for a given task or trajectory segment but can change between different tasks or segments. This task variable is used to condition the dynamics models by

$$p(\mathbf{z}_{t+1}|\mathbf{z}_t, \mathbf{a}_t, \mathbf{l}) = \mathcal{N}(\mathbf{z}_{t+1}|\mathbf{A}_t\mathbf{z}_t + b(\mathbf{a}_t) + c(\mathbf{l}), \sigma^{\mathrm{dyn}}\mathbf{I}).$$

Similar to the ac-RKN, the HiPRSSM assumes the actions are free of uncertainty. However, they assume the task variable $\mathbf{l}$ to be uncertain, i.e., it is modeled as a Gaussian random variable with mean $\boldsymbol{\mu}_l$ and covariance Σ_l.

Thus, to retain the RKN's and ac-RKN's closed-form inference, the HiPRSSM can still use arbitrary nonlinear functions to model the action conditioning but needs a linear model $c(\mathbf{l}) = \mathbf{C}_t\mathbf{l}$ to incorporate the task uncertainty in closed form. Under those assumptions and an u, the predict step is given by

$$\mathbf{z}_{t+1}^{-} = \mathbf{A}_t\mathbf{z}_t + b(\mathbf{a}_t) + \mathbf{C}\boldsymbol{\mu}_{\mathrm{l}} \text{ and } \Sigma_{t+1}^{-} = \mathbf{A}_t\Sigma_t\mathbf{A}_t^T + \mathbf{C}\Sigma_l\mathbf{C}^T + \Sigma^{\mathrm{dyn}}. \tag{4.1}$$

However, Shaj et al. (2022) also explore an alternative approach and use a nonlinear model $c(\mathbf{l})$. While the resulting predict step for the mean is still straightforward, uncertainty propagation is no longer possible in closed form. Inspired by Equation 4.1, they introduce a second nonlinear function $c_\sigma(\Sigma_l)$ to transform the task covariance and add it to the variance of the propagated state and the one of the system dynamics.

$$\mathbf{z}_{t+1}^{-} = \mathbf{A}_t\mathbf{z}_t + b(\mathbf{a}_t) + c(\mathbf{l}) \text{ and } \Sigma_{t+1}^{-} = \mathbf{A}_t\Sigma_t\mathbf{A}_t^T + c_\sigma(\Sigma_l) + \Sigma^{\mathrm{dyn}},$$

where c and c_σ are parameterized by neural networks.

For inference, the HiPRSSM uses a two-stage process. First, the task parameter $\mathbf{l}$ and uncertainty Σ_l is obtained from a context set using a Bayesian context aggregation mechanism (Volpp et al., 2021). In principle, this context set can be anything that describes the current task or conditions. Shaj et al. (2022) use a sliding window over observations and actions. In the second step, the dynamics model is conditioned on the inferred task parameter $\mathbf{l}$, and the latent states are inferred using Kalman filtering in the latent space, following the RKN and ac-RKN approaches.

For evaluation, Shaj et al. (2022) first repeat their prior experiments with the ac-RKN on the musculoskeletal robot (Shaj et al., 2020). The HiPRSSM learns a model that outperforms the ac-RKN, indicating there are non-stationary effects in the data, even though care was taken to ensure consistent conditions during the original data collection. They further evaluate their approach using both real and simulated robot data, which was deliberately collected under challenging and changing conditions. Here, the HiPRSSM learns models that effectively identify meaningful latent task variables while improving prediction performance across tasks. When comparing the linear and nonlinear task conditioning, they find that the nonlinear model performs better in most experiments.

The HiPRSSM was developed concurrently with the method presented in Chapter 5 to tackle an orthogonal problem. However, it further emphasizes the benefits of appropriately structured latent variable models and the importance of uncertainty propagation in the latent space. The benefits of the nonlinear parameterization of the task model $c(\mathbf{l})$ and, in particular, the additional nonlinear function for the uncertainty $c_\sigma(\Sigma_l)$ suggest benefits of more complex uncertainty models. The methods presented in Chapter 5 and Chapter 6 take this idea further by using state and action-dependent transition noise models, which nevertheless still allow closed-form uncertainty propagation over time.

Finally, including ideas from the HiPRSSM into the SSM-based world models and representations introduced in this thesis provides a promising avenue for approaches that generalize better over different domains and tasks. Gospodinov et al. (2024) presents preliminary results in this direction by combining ideas from the HiPRSSM with Dreamer (Hafner et al., 2020) to learn world models that can adapt to changing dynamics and multiple tasks.

Contribution Statement. *VS led the projects, designed the HiPRSSM, including the training objectives, conducted the experiments, and wrote large parts of the paper. DB provided the data for the musculoskeletal robot, helped conducting the corresponding experiments, and provided feedback on the paper. RS implemented the wheeled robot simulator, collected the data, and conducted the corresponding experiments. I assisted in the model design, implementation, and writing the paper. GN supervised the project and provided feedback on the paper and experiment design.*

The work presented in Gospodinov et al. (2024) is the result of EG's Master's thesis supervised by VS. I assisted in the final stages of the project and in writing the article.

4.5.3. Switching Recurrent Kalman Networks

This section summarizes Switching Recurrent Kalman Networks, *published at the Machine Learning for Autonomous Driving Workshop at NeurIPS 2021(Giao Nguyen-Quynh, Philipp Becker, Chen Qiu, Maja Rudolph, and Gerhard Neumann, 2021) and draws connections to the RKN, the motivation behind EIM and the variational objectives for learning deep SSMs presented in Chapter 5.*

The Switching Recurrent Kalman Network (SRKN) (Nguyen-Quynh et al., 2021) extends the RKN to explicitly model multimodal dynamics. Towards this end, they combine the RKN's locally linear dynamics model and efficient inference with the concept of switching SSMs (Murphy, 1998; Ghahramani and Hinton, 2000). Such switching SSMs introduce an additional latent variable $\boldsymbol{\zeta}_t$ that indicates which of several possible dynamics models is active at a given time step. While classical switching SSMs typically use discrete latent variables, the SRKN uses a continuous latent variable $\boldsymbol{\zeta}_t$, which it uses to compute a locally linear transition matrix. Like the RKN, the SRKN parameterizes its transition matrix as a linear combination of K basis matrices $\mathrm{A}^{(k)}$. However it computes the coefficients α around the additional latent variable $\boldsymbol{\zeta}_t$ instead of the posterior mean $\boldsymbol{\mu}_t^+$, i.e.,

$$\mathrm{A}(\boldsymbol{\zeta}_t) = \sum_{k=1}^{K} \alpha\left(\boldsymbol{\zeta}_t^{(k)}\right)\mathrm{A}^{(k)} \text{ with } \left(\alpha(\boldsymbol{\zeta}_t^{(1)}), \cdots, \alpha(\boldsymbol{\zeta}_t^{(K)})\right) = \operatorname{softmax}(\boldsymbol{\zeta}_t).$$

With this matrix, the SRKNs dynamics model is given by

$$p(z_{t+1}|\mathbf{z}_t, \boldsymbol{\zeta}_t) = \mathcal{N}\left(\mathbf{z}_{t+1}|\mathbf{A}(\boldsymbol{\zeta}_t)\mathbf{z}_t, \sigma^{\text{dyn}}\mathbf{I}\right),$$

where σ^{dyn} is a scalar noise parameter and $\mathbf{I}$ is the identify matrix. Additionally, they assume the standard SSM observation model $p(\mathbf{o}_t|\mathbf{z}_t)$ and reuse the RKNs parameterization. In particular, following the switching SSM literature, they model the observations as independent of the switching variable $\boldsymbol{\zeta}_t$. To complete the specification of their generative transition models, Nguyen-Quynh et al. (2021) model the dynamics of the switching latent variable $\boldsymbol{\zeta}_t$ as

$$p(\boldsymbol{\zeta}_{t+1}|\boldsymbol{\zeta}_t, \mathbf{z}_\text{t}) = \mathcal{N}\left(\boldsymbol{\mu}_{\zeta_t}, \Sigma_{\zeta_t}\right) \text{ with } (\boldsymbol{\mu}_{\zeta_t}, \Sigma_{\zeta_t}) = f(\mathbf{h}_t, \mathbf{z}_{t-1}) \text{ and } \mathbf{h}_t = \text{GRU}(\boldsymbol{\zeta}_{t-1}, \mathbf{h}_{t-1}).$$

Similar to the Recurrent State Space Model (RSSM) (Hafner et al., 2019) (Section 2.2.2.1), this model combines a gated recurrent unit (GRU) (Cho et al., 2014) with a small neural network f to model dynamics in a latent space. However, the SRKN uses this for modeling the switching latent variable $\boldsymbol{\zeta}_t$ instead of the latent state $\mathbf{z}_t$ directly.

To capture truly multimodal behavior, the SRKN cannot use the RKN's direct likelihood-based training objective but needs to use a variational objective for SSMs similar to those discussed in Section 2.2.2. To this end, the SRKN uses a variational distribution of the form

$$q(\mathbf{z}_{\leq T}, \boldsymbol{\zeta}_{\leq T}|\mathbf{o}_{\leq T}) = \prod_{t=1}^{T} q(\mathbf{z}_t|\boldsymbol{\zeta}_t, \mathbf{z}_{t-1}, \mathbf{o}_t)q(\boldsymbol{\zeta}_t|\boldsymbol{\zeta}_{\leq t-1}, \mathbf{o}_t).$$

Here, the dependency of the switching variable $\boldsymbol{\zeta}_t$ on the previous latent state $\mathbf{z}_{t-1}$ purposefully omitted following previous works (Yingzhen and Mandt, 2018) to empirically avoid averaging problems. The first term is formed with Kalman filtering, following the RKN, and the second term is learned by a neural network with another GRU. However, with this inference model, Nguyen-Quynh et al. (2021) rely on a filtering inference scheme, which can lead to suboptimal results, as discussed in Chapter 5. Additionally, by reusing the RKN to compute $q(\mathbf{z}_t|\boldsymbol{\zeta}_t, \mathbf{z}_{t-1}, \mathbf{o}_t)$ they implicitly marginalize over the previous latent state $\mathbf{z}_{t-1}$, introducing an additional bias in the lower bound.

Nguyen-Quynh et al. (2021) evaluate their approach on both simulated and real-world data and show that it can effectively capture and predict multimodal behavior. Figure 4.6 shows examples of predictions of human taxi driving behavior based on real-world data. While the results indicate that the SRKN can usually capture the multimodal nature of the data well, its predictions are often less accurate than those of related approaches, which is arguably an artifact of the issues with the variational objective discussed earlier. To mitigate this issue, Nguyen-Quynh et al. (2021) add an additional next state prediction loss, akin to the one used by the ac-RKN, to the training objective.

Both the SRKN and the EIM (Chapter 3) address the problem of modeling multimodal human behavior, specifically trying to avoid unsafe averaging over behavioral modes.

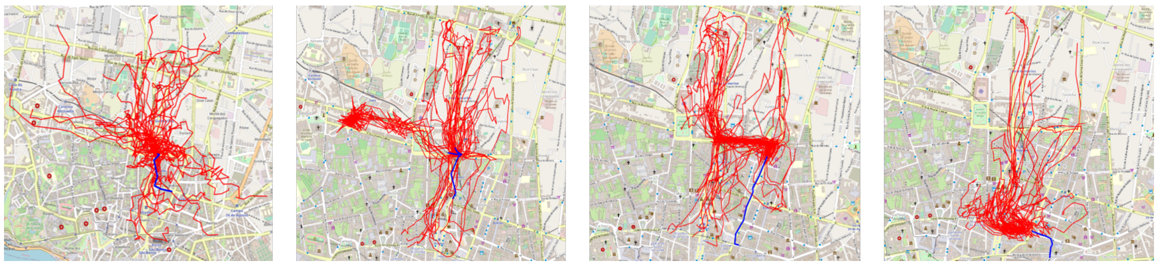


Figure 4.6.: SRKN predictions on test data from the (O'Connell et al., 2015) dataset modeling human driving behavior based on data from taxi drivers in collected in Porto, Portugal. Each subplot shows 50 predicted trajectories (red) given a single observation context (blue). The results show that the SRKN can capture the multimodal nature of human driving behavior. While the trajectories follow the general structure of the underlying road network, they can fail to capture the exact details and predict driving outside of the road. *Figure adapted from Nguyen-Quynh et al. (2021). Caption rewritten.*

However, they differ in their focus and approach. While the SRKN focuses on predicting future behavior, EIM is primarily designed for imitation. Additionally, there is a close relation between switching SSMs and the mixture models used by EIM, as switching SSMs can be seen as a mixture of transition models to capture multimodal behavior. In this context, using a continuous latent switching variable ζ_t to model the switching between different modes can be viewed as a mixture model with infinite components.

Contribution Statement. *The SRKN and the resulting workshop paper are the results of GNQ's Master's thesis supervised by CQ, MR, and myself. GNQ derived and implemented the SRKN, conducted the experiments, and wrote the paper. CQ, MR, and myself provided assistance during the derivations for experimental design and writing of the paper. The basic idea of combining the RKN with a switching latent variable model was developed by MR and myself. GN supervised the project and provided feedback on the paper.*

5. On Uncertainty in Deep State Space Models for Model-Based Reinforcement Learning

Section 2.3.3 discussed the importance of world models for model-based Reinforcement Learning (RL) that not only accurately represent the world but also capture and disentangle uncertainties. In this chapter, we address this challenge by starting to answer the second research question of this thesis

Q2 How can we build a computationally efficient world model for RL that captures and propagates uncertainties in the world while separating them from uncertainties due to lack of training data?

We start with a critical analysis of prominent world modes for model-based RL, such as the Recurrent State Space Model (RSSM) (Hafner et al., 2020) (Section 2.2.2.1) to answer this question and make two key observations. First, while RSSMs are very successful in deterministic and game-like environments, they struggle in environments with highly uncertain observations, such as those encountered in real-world applications. Second, the RSSM does not explicitly model epistemic uncertainty, defying the conventional wisdom in model-based RL that epistemic and aleatoric uncertainties should be modeled as discussed in Section 2.3.3.1.

Here, our analysis revealed that the reasons behind these issues are connected in intricate and counterintuitive ways. Both are fundamentally caused by the RSSM's simplifying inference scheme, which learns a model based on filtered, not smoothed, belief states. While this might seem like a reasonable simplification at first, we will see that it causes the RSSM to overestimate the aleatoric uncertainty in the model systematically. On the one hand, this overestimation of the aleatoric uncertainty benefits the model in deterministic environments, as it acts as an implicit regularization and compensates for the lack of epistemic uncertainty. On the other hand, it leads to poor performance in environments with highly uncertain observations, as the model cannot learn a good representation of the system dynamics.

To address these shortcomings, we turned to our previous work on the RKN to devise a new, more principled probabilistic world model for model-based RL, the Variational Recurrent Kalman Network (VRKN). First, we extended the RKN from only filtering to a full smoothing approach and replaced the likelihood-based objective with a full autoencoding variational Bayes (Kingma and Welling, 2013) (Section 2.2.1) objective. Second, we equipped the model with explicit epistemic uncertainty to act as explicit regularization. Third, we used the resulting world model to learn behaviors through model-based RL by planning (Hafner et al., 2019) and latent imagination (Hafner et al., 2020).

While the VRKN is a principled way to model and disentangle epistemic and aleatoric uncertainties, it lacks the computational efficiency needed for large-scale model-based RL. We will address this issue in Chapter 6.

The following was published as On Uncertainty in Deep State Space Models for Model-Based Reinforcement Learning *(Becker and Neumann, 2022) in Transactions on Machine Learning Research (TMLR). It was also presented in the Journal Track of the first Reinforcement Learning Conference (RLC 2024). For this chapter, the contents of the original publication were restructured, and the wording and formulation were slightly revised in several places.*

5.1. Introduction

Accurate system models are crucial for model-based control and Reinforcement Learning (RL) in autonomous systems applications under partial observability. Practitioners commonly use State Space Model (SSMs) (Murphy, 2012) to formalize such systems. SSMs consist of a dynamics model, describing how one state relates to the next, and an observation model, which describes how system states generate observations. Yet, dynamics and observation models are unknown for most relevant problems, and exact inference in the resulting SSM is usually intractable. Researchers have proposed numerous approaches to learning the models from data and approximating the inference to solve those issues.

Recurrent State Space Models (RSSMs) (Hafner et al., 2019) are of particular interest here. Using RSSMs as the backbone for their Deep Planning Network (PlaNet), Hafner et al. (2019) showed that variational latent dynamics learning can succeed in image-based RL for complex control tasks. Combined with simple planning, RSSMs can match the performance of model-free RL while requiring significantly fewer environment interactions. The authors later improved upon their original model, including a parametric policy trained on imagined trajectories (Dreamer) (Hafner et al., 2020). In general, approaches based on RSSMs have found considerable interest in the model-based RL community. Yet, while RSSMs draw inspiration from classical SSMs, they use a simplified inference scheme. During inference, they assume the belief is independent of future observations instead of using the correct smoothing assumptions (Murphy, 2012) to obtain the belief. We formalize this observation in Section 5.2 and discuss how these simplified assumptions result in a theoretically looser variational lower bound. Further, we analyze the effects of these assumptions on model learning and find they cause an overestimation of aleatoric uncertainty. Such aleatoric uncertainty stems from partial observability or inherent stochasticity of the system (Hüllermeier and Waegeman, 2021). It plays an important role in many realistic tasks, e.g., in the form of occlusions, missing observations, or observations arriving in different modalities and frequencies. Consider, for example, low-frequency camera images providing external information and high-frequency proprioceptive measurements of the robot's internal state. Given such sensory inputs, we require an appropriate estimation of aleatoric uncertainty to fuse all information optimally and form accurate belief states. Our experiments show that RSSMs perform sub-optimally in such cases, which we attribute to the wrong estimation of aleatoric uncertainty.

Furthermore, we argue that the RSSM's inference assumptions are a double-edged sword for model-based RL. On the one hand, the overestimated aleatoric uncertainty can be beneficial as it leads to dynamics models that generalize better and are more robust to objective mismatch (Luo et al., 2019; Lambert et al., 2020). Such objective mismatch arises because the model aims to maximize the data likelihood while the metric we care about is the agent's reward. Additionally, as the agent explores during training, it will encounter states and actions the model has not seen, causing a distribution shift between training and data collection. While many approaches rely on explicit epistemic uncertainty to tackle this issue (Chua et al., 2018; Janner et al., 2019), RSSMs succeed without capturing epistemic uncertainty. Such epistemic uncertainty (Hüllermeier and Waegeman, 2021) stems from the lack of training data and plays an important role in model-based RL, where the data is not given but collected during the training process. On the other hand, this heuristic approach to address objective mismatch complicates the design and analysis of the approach as the robustness is a side-product of the inference scheme and does not follow well-motivated principles. For example, purely stochastic versions of the RSSM give unsatisfactory results, and it is unclear why this is the case. Arguably, an explanation could be the overestimated aleatoric uncertainty causing the purely stochastic model to be unstable. As a remedy, Hafner et al. (2019) introduce a deterministic path, i.e., they combine stochastic and deterministic features to form the belief state.

We show that removing this issue for RSSMs by implementing a naive approach to smoothing yields unsatisfactory results, even if the model uses explicit epistemic uncertainty to compensate for the missing overestimation of the aleatoric uncertainty. To make smoothing inference work, we redesign the model using well-understood and theoretically founded components to model aleatoric and epistemic uncertainty. We dub the resulting model Variational Recurrent Kalman Network (VRKN). It uses a Linear Gaussian State Space Model embedded in a latent space, which allows closed-form smoothing inference and proper estimation of aleatoric uncertainty. Furthermore, as the inference builds on Bayes rule, it provides a natural treatment of missing observations and sensor fusion settings. Additionally, the VRKN uses Monte Carlo Dropout (Gal and Ghahramani, 2016) for a Bayesian treatment of its transition model's parameters, explicitly modeling epistemic uncertainty.

We first show that the resulting architecture allows model-based agents to perform comparably to RSSM-based agents on standard benchmarks from the Deep Mind Control Suite (Tassa et al., 2018). Those environments are almost deterministic, i.e., the aleatoric uncertainty is low. Subsequently, we modify tasks from the Deep Mind Control Suite to create three different scenarios that exhibit high aleatoric uncertainty, i.e., we introduce occlusions, missing observations, and sensor fusions problems of external camera images and internal proprioceptive feedback which are available at different frequencies. Using those, we show that the VRKN improves the agents' performance due to its accurate

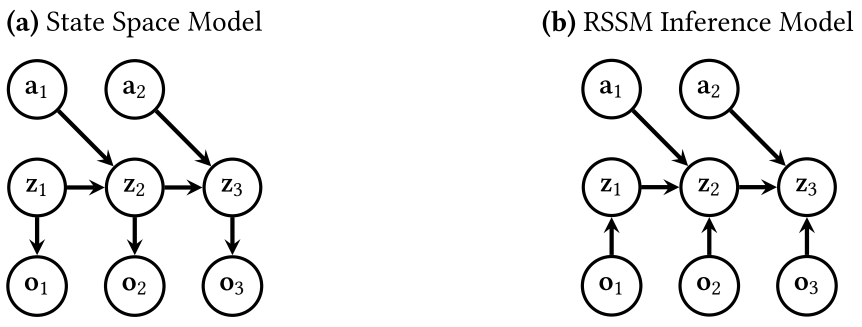


Figure 5.1.: **(a)** State space model, serving as the generative model throughout this work. **(b)** Graphical Model underlying the RSSM (Hafner et al., 2019) inference scheme. In contrast to the generative SSM, the direction between observations and latent states is inverted. These independence assumptions result in a simplified inference and subtle effects on model learning.

estimation of aleatoric uncertainty. Our approach builds on well-motivated components for aleatoric and epistemic uncertainty. Additionally, it does not require heuristic model components such as a deterministic path. As such, it may serve as a basis for future research into improving SSMs for model-based RL.

To summarize our contributions:

1. We analyze the assumptions underlying the RSSM and show that they are not only suboptimal but also have subtle effects on model learning. They lead to an overestimation of the aleatoric uncertainty (Section 5.2.1 and Equation 5.2.2).
2. We argue that, counterintuitively, this suboptimal inference is beneficial for the RSSMs performance as it addresses objective mismatch in a heuristic way (Section 5.2.3). We evaluate several modifications to the RSSM to provide empirical evidence to support this hypothesis (Section 5.4.1).
3. We introduce the VRKN, providing a more principled inductive bias for smoothing inference than the RSSM (Section 5.3). We again show that a smoothing inference without additional measures results in suboptimal performance, yet, when combined with epistemic uncertainty, the VRKN's improved inductive bias allows it to close the performance gap on standard benchmarks (Section 5.4.1).
4. We show that VRKN-based agents improve performance in tasks where correct uncertainty estimation matters. Here we consider tasks with partial observability, missing information, or tasks that require sensor fusion (Section 5.4.2).

5.2. Inference and Learning in State Space Models

State Space Models (SSMs) (Murphy, 2012) assume that a sequence of observations $\mathbf{o}_{\leq T} = (\mathbf{o}_t)_{t=0\cdots T}$ is generated by a sequence of latent state variables $\mathbf{z}_{\leq T} = (\mathbf{z}_t)_{t=0\cdots T}$, given a sequence of actions $\mathbf{a}_{\leq T} = (\mathbf{a}_t)_{t=0\cdots T}$. In SSMs, each observation $\mathbf{o}_t$ is assumed to only depend on the current latent state $\mathbf{z}_t$ via an observation model $p(\mathbf{o}_t|\mathbf{z}_t)$. Further, they assume the latent states are Markovian, i.e., each latent state only depends on its direct predecessor and the corresponding action via a dynamics model $p(\mathbf{z}_{t+1}|\mathbf{z}_t, \mathbf{a}_t)$. Finally, the initial state is distributed according to a distribution $p(\mathbf{z}_0)$. Figure 5.1 shows the corresponding graphical model. Typically, when inferring latent states from observations, we consider three different beliefs for each $\mathbf{z}_t$. Those are the prior $p(\mathbf{z}_t|\mathbf{o}_{\leq t-1}, \mathbf{a}_{\leq t-1})$, i.e., the belief before observing $\mathbf{o}_t$, the posterior, $p(\mathbf{z}_t|\mathbf{o}_{\leq t}, \mathbf{a}_{\leq t-1})$, i.e., the belief after observing $\mathbf{o}_t$, as well as the smoothed belief $p(\mathbf{z}_t|\mathbf{o}_{\leq T}, \mathbf{a}_{\leq T})$ which is conditioned on all future observations and actions, until the last time step T. We refer to those estimates as state-beliefs to distinguish them from dynamics distributions, which are conditioned on the previous state $\mathbf{z}_{t-1}$ such as the prior dynamics $p(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{a}_{t-1})$, the posterior dynamics $p(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{o}_t, \mathbf{a}_{t-1})$ and the smoothed dynamics $p(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{o}_{\geq t}, \mathbf{a}_{\geq t-1})$. To get from a dynamics distribution to the corresponding state belief we need to marginalize out the previous state, e.g.,

$$p(\mathbf{z}_t|\mathbf{o}_{\leq t-1}, \mathbf{a}_{\leq t-1}) = \int p(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{a}_{t-1}) p(\mathbf{z}_{t-1}|\mathbf{o}_{\leq t-1}, \mathbf{a}_{\leq t-1}) d\mathbf{z}_{t-1}$$

for the prior.

Traditionally, the independence assumptions of the generative model, shown on the left side of Figure 5.1, are also used for inference. Yet, RSSMs (Hafner et al., 2019) work with a different set of assumptions during inference, shown on the right side of Figure 5.1[1]. In particular, in this graphical model the state $\mathbf{z}_t$ is conditionally independent of all observations $\mathbf{o}_{>t}$ and actions $\mathbf{a}_{\geq t}$ given $\mathbf{z}_{t-1}$, $\mathbf{a}_{t-1}$, and $\mathbf{o}_t$, which is not the case for the standard SSM. We discuss the effects of those assumptions on inference and model learning. We refer to Appendix C.1 for more detailed derivations of the following identities.

5.2.1. Variational Inference for State Space Models

Inferring beliefs over the latent states given observations and actions is intractable for most models. Thus, we usually use approximate inference methods. In this work, we focus on Variational Inference. Such variational methods introduce an auxiliary distribution q over the latent variables given the observable variables. They decompose the data log-likelihood into a variational lower bound and a Kullback-Leibler divergence (KL) term, measuring how close the bound is to the data log-likelihood. For State Space Models (SSMs) this decomposition is given as

$$\begin{aligned} &\log p(\mathbf{o}_{\leq T}|\mathbf{a}_{\leq T}) \qquad (5.1)\\ &= \underbrace{\mathbb{E}_{q(\mathbf{z}_{\leq T}|\mathbf{o}_{\leq T}, \mathbf{a}_{\leq T}))}\left[\log \frac{p(\mathbf{o}_{\leq T}, \mathbf{z}_{\leq T}|\mathbf{a}_{\leq T})}{q(\mathbf{z}_{\leq T}|\mathbf{o}_{\leq T}, \mathbf{a}_{\leq T})}\right]}_{\text{Lower Bound}} + \underbrace{\mathrm{KL}\left(q(\mathbf{z}_{\leq T}|\mathbf{o}_{\leq T}, \mathbf{a}_{\leq T}) || p(\mathbf{z}_{\leq T}|\mathbf{o}_{\leq T}, \mathbf{a}_{\leq T})\right)}_{\text{KL term}(\geq 0)}. \end{aligned}$$

[1] The full model of Hafner et al. (2019) also includes a deterministic-path which is of no concern regarding the discussion here. Thus, we omit it for brevity. Furthermore, Hafner et al. (2019) compare their RSSM to a baseline they abbreviate as SSM (stochastic state model), which also builds on the simplified assumptions. In this work, we refer to all approaches based on the simplified assumptions as RSSM.

Variational inference methods try to find the optimal q by maximizing the lower bound or minimizing the KL term. If a q can be found such that $q(\mathbf{z}_{\leq T}|\mathbf{o}_{\leq T}, \mathbf{a}_{\leq T}) = p(\mathbf{z}_{\leq T}|\mathbf{o}_{\leq T}, \mathbf{a}_{\leq T})$, the KL term is 0, and the bound is said to be tight. While the decomposition is valid for arbitrary distributions $q(\mathbf{z}_{\leq T}|\mathbf{o}_{\leq T}, \mathbf{a}_{\leq T})$, we need to pose a set of independence assumptions to obtain tractable variational models. If we again use the independence assumptions of the generative model, shown in Figure 5.1, we can obtain the inference model by explicitly inverting the generative direction using Bayes rule

$$q(\mathbf{z}_{\leq T}|\mathbf{o}_{\leq T}, \mathbf{a}_{\leq T}) = \frac{1}{p(\mathbf{o}_{\leq T})} p(\mathbf{z}_0) \prod_{t=1}^{T} p(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{a}_t) \prod_{t=0}^{T} p(\mathbf{o}_t|\mathbf{z}_t).$$

For certain model parametrizations, this variational distribution can be computed analytically, e.g., by Kalman smoothing if the model is linear and Gaussian. In this case, the KL term vanishes, and the bound is tight.

Yet, RSSMs, as introduced in (Hafner et al., 2019), assume the variational distribution factorizes as

$$q(\mathbf{z}_{\leq T}|\mathbf{o}_{\leq T}, \mathbf{a}_{\leq T}) = q(\mathbf{z}_0|\mathbf{o}_0) \prod_{t=1}^{T} q(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{o}_t, \mathbf{a}_{t-1}).$$

This assumption results in a simplified inference procedure, as the belief over $\mathbf{z}_t$ is assumed to be independent of all future observations $\mathbf{o}_{>t}$, given $\mathbf{z}_{t-1}$, $\mathbf{o}_t$, and $\mathbf{a}_{t-1}$.

Inserting these assumptions into the KL term yields

$$\begin{aligned} &\mathrm{KL}\left(q(\mathbf{z}_{\leq T}|\mathbf{o}_{\leq T}, \mathbf{a}_{\leq T}) || p(\mathbf{z}_{\leq T}|\mathbf{o}_{\leq T}, \mathbf{a}_{\leq T})\right) \\ &= \sum_{t=1}^{T} \mathbb{E}_{q(\mathbf{z}_{t-1}|\mathbf{o}_{\leq t-1}, \mathbf{a}_{\leq t-2})} \left[\mathrm{KL}\left(q(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{o}_t, \mathbf{a}_{t-1}) || p(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{o}_{\geq t}, \mathbf{a}_{\geq t-1})\right)\right]. \end{aligned} \tag{5.2}$$

For general distributions p, the individual terms in this sum will not be 0, as the right-hand side distributions receive information, i.e., future observations and actions, which are ignored by the variational distribution. Thus, this bound can in general not be tight, not even for linear Gaussian models as the resulting variational distribution can in general not represent $p(\mathbf{z}_{\leq t}|\mathbf{o}_{\leq T}, \mathbf{a}_{\leq T})$. Typically, tight variational bounds are preferable as they allow for faster optimization of the marginal log-likelihood.

The above discussion may be hypothetical, as all considered architectures do not provide tight lower bounds due to the use of deep neural networks, which prevents analytic solutions for inference. Still, as a tight lower bound does not even theoretically exist for the RSSM assumptions, we believe this is already an indication of the misspecification of its inference distribution.

5.2.2. Model Learning under Different Inference Assumptions

In the model-based RL setting considered in this work, the generative model is usually assumed to be unknown and we jointly learn a parametric generative model p and an inference model q using an auto-encoding variational Bayes approach (Kingma and Welling, 2013; Sohn et al., 2015) by maximizing the lower bound part of Equation 5.1. Inserting the RSSM-assumptions into the general lower bound gives the objective introduced by Hafner et al. (2019),

$$\begin{aligned}\mathcal{L}_{\text{rssm}}(\mathbf{o}_{\leq T}, \mathbf{a}_{\leq T}) = \sum_{t=1}^{T} & \mathbb{E}_{q(\mathbf{z}_t|\mathbf{o}_{\leq t}, \mathbf{a}_{\leq t})}\left[\log p(\mathbf{o}_t|\mathbf{z}_t)\right] \\ & - \mathbb{E}_{q(\mathbf{z}_{t-1}|\mathbf{o}_{\leq t-1}, \mathbf{a}_{\leq t-1})}\left[\mathrm{KL}\left(q(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{a}_{t-1}, \mathbf{o}_t)||p(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{a}_{t-1})\right)\right].\end{aligned} \tag{5.3}$$

If one instead uses the same factorization assumptions as the generative model, the inference distribution factorizes as

$$q(\mathbf{z}_{\leq T}|\mathbf{o}_{\leq T}, \mathbf{a}_{\leq T}) = q(\mathbf{z}_0|\mathbf{o}_{\leq T}, \mathbf{a}_{\leq T}) \prod_{t=1}^{T} q(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{a}_{\geq t-1}, \mathbf{o}_{\geq t}).$$

Inserting this factorization into the general lower bound leads to

$$\begin{aligned}\mathcal{L}_{\text{ssm}}(\mathbf{o}_{\leq T}, \mathbf{a}_{\leq T}) = \sum_{t=1}^{T} & \mathbb{E}_{q(\mathbf{z}_t|\mathbf{o}_{\leq T}, \mathbf{a}_{\leq T})}\left[\log p(\mathbf{o}_t|\mathbf{z}_t)\right] \\ & - \mathbb{E}_{q(\mathbf{z}_{t-1}|\mathbf{o}_{\leq T}, \mathbf{a}_{\leq T})}\left[\mathrm{KL}\left(q(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{a}_{\geq t-1}, \mathbf{o}_{\geq t})||p(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{a}_{t-1})\right)\right].\end{aligned} \tag{5.4}$$

Opposed to the RSSM-objective, this objective uses the smoothed belief-state $q(\mathbf{z}_t|\mathbf{o}_{\leq T}, \mathbf{a}_{\leq T})$ and the smoothed dynamics $q(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{a}_{\geq t-1}, \mathbf{o}_{\geq t})$. The variational distribution considers all future information until $t = T$ and thus can theoretically yield a tight bound.

These differences have interesting effects on the learned transition model $p(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{a}_{t-1})$. For the RSSM, the belief over past states can, by definition, not change due to additional observations. Thus, any discrepancy between this past belief and the current observation must be explained by the transition model. In contrast, a smoothing inference can also explain the discrepancy by propagating information from observations to past beliefs. In Equation 5.3, this observation is reflected in the expected KL-term, where the expectation does not consider $\mathbf{o}_t$ or other future observations. Thus, the transition model has to explain the transitions from a given $\mathbf{z}_{t-1}$ to $\mathbf{z}_t$, even if $\mathbf{z}_{t-1}$ would be rendered implausible by a future observation.

For a thought experiment illustrating these effects, consider the following scenario. You meet a person holding a box, and they tell you there is a hamster inside. As your prior experience is that people are usually trustworthy, you chose to believe them. Next, the box opens, and a cat jumps out. As you trust your eyes, you now believe it is a cat. Yet, under the RSSM-assumptions, you cannot revise your belief of the first time step and thus still believe it originally was a hamster. When updating your model based on this interaction, you would learn that hamsters can turn into cats, as you cannot capture the arguably more likely explanation that the person lied. Learning under these assumptions requires you to model unlikely events as more likely than they are. Thus, you overestimate the aleatoric uncertainty in the world.

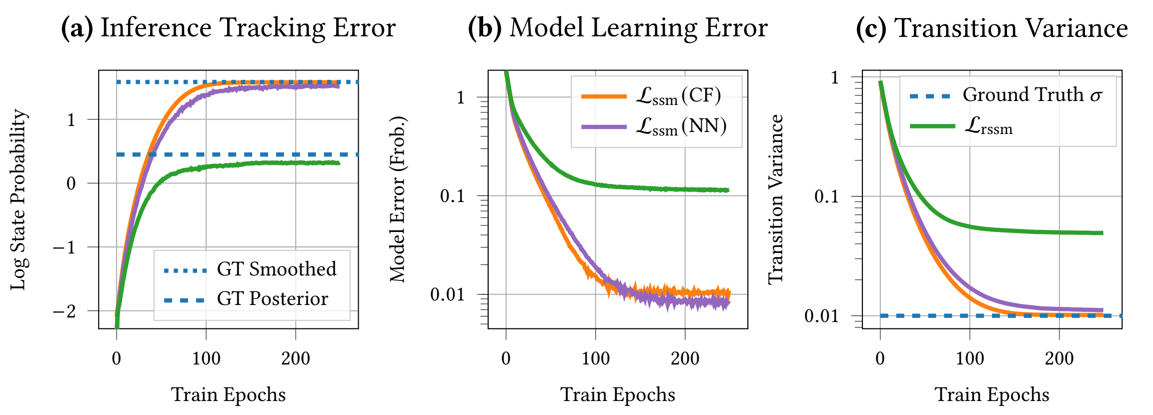


Figure 5.2.: Comparison of inference and model learning results on a simple Linear Gaussian State Space Model without actions, under both RSSM and SSM inference assumptions. We report average results over 10 seeds and error bars indicating 95% bootstrapped confidence intervals. Note that the error bars are too small to be visible. We consider both a closed-form (CF) version based on Kalman Smoothing and a version with a neural network (NN) as an inference model for the SSM-based approaches and find that only the objective matters, not the parametrization of the inference model. **(a)** Log-probability of the ground truth states under the learned models. We compare against the quality of the ground truth smoothed (GT Smoothed) and posterior (GT Posterior) beliefs, computed using a Kalman Smoother and ground truth generative model. While the SSM objective reaches the quality of the smoothed belief, the RSSM-based inference fails to attain the quality of even the posterior belief. **(b)**: Distance between ground truth transition matrix and learned transition matrix, measured using the Frobenius norm. Here, the SSM inference yields a model that is an order magnitude closer to the ground-truth model than that learned by the RSSM-bound. **(c)**: Transition variance $\sigma\mathbf{I}$. With the SSM bound we recover the ground-truth aleatoric uncertainty, yet with the RSSM bound the aleatoric uncertainty is significantly overestimated.

More formally, we can demonstrate this effect using a simple Linear Gaussian State Space Model without actions. We will use a state dimension of 4 and the ground-truth model given by

$$p(\mathbf{z}_0) = \mathcal{N}(\mathbf{z}_0|\mathbf{0}, \mathbf{I}), \quad p(\mathbf{o}_t|\mathbf{z}_t) = \mathcal{N}(\mathbf{o}_t|\mathbf{I}\mathbf{z}_t, 0.025\mathbf{I}), \quad p(\mathbf{z}_{t+1}|\mathbf{z}_t) = \mathcal{N}(\mathbf{z}_{t+1}|\mathbf{A}_{\text{gt}}\mathbf{z}_t, \sigma_{\text{gt}}\mathbf{I})$$

where $\mathbf{I}$ denotes the identity matrix and $\sigma_{\text{gt}} = 0.01$. The transition matrix $\mathbf{A}_{\text{gt}}$ is given as

$$\mathbf{A}_{\text{gt}} = \begin{pmatrix} 1.0 & 0.0 & 0.2 & 0.0 \\ 0.0 & 1.0 & 0.0 & 0.2 \\ -0.2 & 0.0 & 0.95 & 0.0 \\ 0.0 & -0.2 & 0.0 & 0.95 \end{pmatrix},$$

which induces a slightly damped, oscillating behavior.

Using this ground truth model, we generate 1,000 sequences of length 50 and fit a generative model transition model. For this model, we fix the initial state and observation model and only learn a linear dynamics model. This model consists of a 4×4 transition matrix $\mathbf{A}$, which is initialized randomly, sampled from $\mathcal{N}(0, 0.05)$, and an isotropic covariance which we scale by a learned factor σ, initialized with 1.

Even in this simple setting, computing the optimal inference distribution for the RSSM-bound (Equation 5.3) is impossible, and we thus resort to numerical methods. We thus parameterize it as

$q(\mathbf{z}_{t+1}|\mathbf{z}_t, \mathbf{o}_t) = \mathcal{N}(\mathbf{C}_t\mathbf{z}_t + \mathbf{c}_t, \Sigma_t^c)$, where $\mathbf{C}_t, \mathbf{c}_t, \Sigma_t^c$ are emitted by a small neural network given the observation and mean of the previous belief. This network is trained jointly with the generative model. For the state space assumptions, we can compute the optimal inference distribution in closed form using Kalman Smoothing. Thus we do not need an additional inference model. We refer to this approach as $\mathcal{L}_{\text{ssm}}(\text{CF})$ in Figure 5.2. Yet, we additionally consider a NN-based approach to ensure the observed effects are a result of the loss used, not the parameterization of the model. To this end, we use a GRU (Cho et al., 2014) to process the observations backward, extracting a latent representation of all future observations $\mathbf{o}_{\geq t}$ for each time step. We then feed this representation into the NN used for $\mathcal{L}_{\text{rssm}}$ described above to get a NN emitting parameters for $q(\mathbf{z}_{t+1}|\mathbf{z}_t, \mathbf{o}_{\geq t})$ and refer to the resulting model as $\mathcal{L}_{\text{ssm}}(\text{NN})$.

Figure 5.2 summarizes the results and demonstrates that the RSSM-bound leads to a suboptimal inference and consequently to learning a wrong model. In particular, we can see that for the RSSM, the transition variance $\tilde{\sigma}$ is much larger than the ground-truth value $\sigma = 0.01$ as all unexpected observations have to be explained by the transitions instead of correcting the beliefs of past time steps. Additionally, we can see that $\mathcal{L}_{\text{ssm}}(\text{CF})$ and $\mathcal{L}_{\text{ssm}}(\text{NN})$, behave very similar across all considered metrics, which demonstrates that the loss is the determining factor, not the parameterization.

5.2.3. The Interplay of Policy Optimization and Regularization

Despite the theoretical considerations in the previous section, RSSMs work well for model-based RL. It is well known that model-based RL suffers from an objective mismatch (Luo et al., 2019; Lambert et al., 2020) issue. This issue arises because the model aims to maximize the ELBO (Equation 5.1) but is evaluated based on the agent's reward. The effect is further amplified by the distribution shift between training and data collection, as data collection is typically performed only after a policy improvement step. In RL, we explicitly want the agent to explore unseen parts of the state-action space, encountering observations the model has not seen before. Thus, training the underlying model requires careful regularization such that wrong predictions do not prevent the agent from exploring relevant parts of the state space. Many model-based RL approaches (Chua et al., 2018; Janner et al., 2019) handle this issue by explicitly modeling the epistemic uncertainty of the model, which is not required by the RSSM. Instead, we argue that RSSMs rely on the overestimated aleatoric uncertainty caused by suboptimal inference to address objective mismatch in a more heuristic manner. The overestimation implicitly regularizes the RSSM as it forces the transition model to model unlikely events with higher probability. This regularization thus implicitly prevents overconfident model predictions due to overfitting. Yet, while it alleviates the objective mismatch issue, there are also drawbacks to this heuristic solution. First, it complicates the model design, analysis, and improvement of RSSMs. As already observed by Hafner et al. (2019), a fully stochastic model based on the RSSM-assumptions underperforms without additional measures as it fails at reliably propagating information for multiple time steps. As a remedy, Hafner et al. (2019) introduces a deterministic path, i.e., a Gated Recurrent Unit (Cho et al., 2014), and base the belief update on this instead of the stochastic belief. Second, as we show in Section 5.4.2, there are settings where appropriately capturing the aleatoric uncertainty is important, and failing to do so can hurt performance.

In a first attempt to address those issues, we minimally adapt the RSSM to be capable of smoothing. This Smoothing RSSM uses a GRU which is added before the actual RSSM and runs backwards over

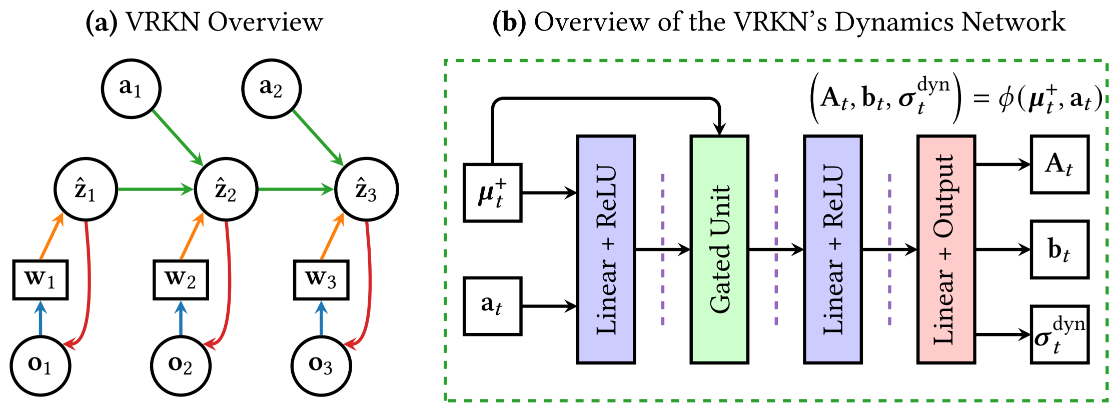


Figure 5.3.: **(a)** We use an encoder (blue) to extracts an intermediate representation $\mathbf{w}_t$ and the observation noise $\boldsymbol{\sigma}_t^{\mathrm{w}}$ from the original observation $\mathbf{o}_t$. Both, $\mathbf{w}_t$ and $\boldsymbol{\sigma}_t^{\mathrm{w}}$, are then used to update the state estimate $\hat{\mathbf{z}}_t$ using the closed-form Kalman belief update (Bayes rule for Gaussians) (orange). A decoder (red) reconstructs the observation given a state sample. We propagate the latent state estimate to the next time step using the transition model $p(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{a}_{t-1})$ (green). The parameters of this distribution, $\mathbf{A}_t$, $\mathbf{b}_t$, and $\boldsymbol{\sigma}_t^{\mathrm{dyn}}$, are computed using the network $\phi(\boldsymbol{\mu}_t^+, \mathbf{a}_t)$, shown in **(b)**. The network receives the current posterior mean $\boldsymbol{\mu}_t^+$ and the action $\mathbf{a}_t$ as inputs. Thus, the resulting dynamics are linearized around the posterior mean, which allows closed-form belief propagation, similar to an extended Kalman filter. For stability during training, the network includes a gated unit, i.e., a GRU cell, which receives the posterior mean $\boldsymbol{\mu}_t^+$ at the memory input. For a Bayesian treatment of the transition model's parameters, we include Monte Carlo Dropout layers at the positions indicated by the purple dashed lines.

the representations extracted by the encoder, effectively accumulating all future information. The RSSM then receives the output of this GRU instead of the original observation as input. This model is an extension of the neural network based inference model for $\mathcal{L}_{\mathrm{ssm}}$, introduced in the previous section ($\mathcal{L}_{\mathrm{ssm}}$(CF) in Figure 5.2). We compare this approach to the original RSSM and present the results in Section 5.4.1, in particular Figure 5.4. Those results show that the performance decreases compared to the original RSSM. We argue that with a proper inference, the overestimated aleatoric uncertainty no longer regularizes the model, making it more prone to objective mismatch. Following other methods, we try to improve the results by modeling epistemic uncertainty. To this end, we use Monte Carlo Dropout (MCD) but find that it does not help to improve the Smoothing RSSM's performance. From these results, again presented in Figure 5.4, we conclude that solely addressing the suboptimal inference assumption is insufficient, but we also need to rethink the model's parameterization. We postulate that the additional GRU for the backward pass is a poor inductive bias and that we require a smoothing approach that adds as little complexity as possible to the model. In the next section, we will introduce an architecture that allows for parameter-free smoothing.

5.3. Variational Recurrent Kalman Networks

To provide a theoretically better-grounded alternative to the RSSM, we require a model which allows tractable inference while still scaling to complex image-based control tasks. Further, the architecture should allow efficient computation of smoothed and posterior state beliefs and dynamics. While we need smoothed beliefs for training, we require (filtered) posteriors for online control. In particular, as only the smoothed beliefs are explicitly used for training, the model needs to provide a strong inductive bias that enables it to still produce reasonable posterior estimates.

To meet these criteria, we introduce a new parametrization of the latent dynamics based on a Linear Gaussian State Space Model (LGSSM) (Murphy, 2012) embedded in a latent space. The linear Gaussian assumptions allow a rigorous treatment of uncertainties while working in a learned latent space allows for modeling high-dimensional and non-linear systems. Inference in this LGSSM amounts to (extended) Kalman filtering and smoothing, enabling smoothing inference without introducing additional parameters besides the latent observation and dynamics model. Due to this property, the architecture can perform proper smoothing inference for training and also produce reasonable posterior beliefs for online control. As an additional benefit, the formulation can naturally handle missing observations and the fusion of multiple sensors, making it amenable to many realistic problems. To include epistemic uncertainty in our approach, we use Monte Carlo Dropout (Gal and Ghahramani, 2016) for a Bayesian treatment of the LGSSM's transition model. We name the resulting approach Variational Recurrent Kalman Network (VRKN). Figure 5.3 shows a schematic overview.

In the following, we first introduce the VRKN's dynamics model, which is shared between the inference and generative parts of the model. Next, we introduce the parameterization of the inference model q and the generative model p. We describe how to train the model and use it for control and conclude by elaborating on the natural fusion mechanism.

5.3.1. The VRKN's Dynamics Model

Both, the VRKN's inference model q and the generative model p, use the same dynamics model. Such parameter sharing is common for variational time-series models (Karl et al., 2016; Fraccaro et al., 2017; Hafner et al., 2019; Klushyn et al., 2021), as it simplifies the architecture, reduces the number of parameters, and simplifies training.

We model the latent dynamics as

$$p(\mathbf{z}_{t+1}|\mathbf{z}_t, \mathbf{a}_t) = \mathcal{N}\left(\mathbf{z}_{t+1}|\mathbf{A}_t(\boldsymbol{\mu}_t^+, \mathbf{a}_t)\mathbf{z}_t + \mathbf{b}_t(\boldsymbol{\mu}_t^+, \mathbf{a}_t), \boldsymbol{\sigma}_t^{\text{dyn}}(\boldsymbol{\mu}_t^+, \mathbf{a}_t)\right),$$

where $\boldsymbol{\mu}_t^+$ denotes the mean of the posterior state estimate $q(\mathbf{z}_t|\mathbf{o}_{\leq t}, \mathbf{a}_{\leq t})$ provided by the inference model. Thus, we learn a linearized dynamics around the current posterior mean. This parametrization builds on the work of Shaj et al. (2020), who propose using a model that is locally linear in the state while depending non-linearly on the action. Building on the assumption that there is no uncertainty in the actions, it is maximally flexible while still allowing the usage of extended Kalman filtering (Jazwinski, 1970). Here, the linearization around the current posterior mean $\boldsymbol{\mu}_t^+$ enables closed-form propagation of state beliefs.

We model $\mathbf{A}_t$ to be a diagonal matrix which is emitted together with the offset term $\mathbf{b}_t$ and the transition noise $\boldsymbol{\sigma}_t^{\text{dyn}}$ by a single neural network $\left(\mathbf{A}_t, \mathbf{b}_t, \boldsymbol{\sigma}_t^{\text{dyn}}\right) = \phi(\boldsymbol{\mu}_t^+, \mathbf{a}_t)$. We carefully design this network to prevent the state estimates and gradients from growing indefinitely during training. First, as $\mathbf{A}_t$ is diagonal, its values are its eigenvalues, and constraining them in an appropriate range ensures stable dynamics. To this end, we use an activation of the form $f(x) = s \cdot \text{sigmoid}(x + b) + m$ where we choose s, b, and m such that it saturates at 0.1 and 0.99 while $f(0) = 0.9$. Here, the intuition is that we want plausible and stable dynamics, which we initialize as a slightly dampened system. Second, we employ a gating mechanism to mitigate problems with vanishing and exploding gradients. To this end, we use a standard GRU cell implementation but feed the posterior mean $\boldsymbol{\mu}_t^+$ into the memory input, i.e., $\phi(\boldsymbol{\mu}_t^+, \mathbf{a}_t) = \phi_2(\text{GRU}(\phi_1(\boldsymbol{\mu}_t^+, \mathbf{a}_t), \boldsymbol{\mu}_t^+))$. The resulting model is still fully stochastic and linear in $\mathbf{z}_t$. In contrast to the RSSM, it does not use a deterministic path as the GRU cell does not introduce an additional deterministic memory and is used solely for addressing problems arising from unstable dynamics and gradients. We empirically found this helpful and took inspiration from standard recurrent architectures (Hochreiter and Schmidhuber, 1997; Cho et al., 2014), where gating mechanisms are a well-known approach to mitigate instabilities. The right side of Figure 5.3 provides an overview of the dynamics model architecture.

Modeling Epistemic Uncertainty. As discussed in Section 5.2.3, the overestimated aleatoric uncertainty of the RSSM's transition model avoids overconfident estimates due to overfitting and compensates for the lack of explicit epistemic uncertainty. Thus, as our approach captures the aleatoric uncertainty correctly, we need to explicitly consider epistemic uncertainty to obtain a model that is also useable for policy optimization. We use Monte Carlo Dropout (Gal and Ghahramani, 2016) due to its simplicity and include corresponding layers at appropriate points in the transition model. Those point are shown on the right side of Figure 5.3.

5.3.2. Inference Model

We aim for a model that can work with high-dimensional observations which depend non-linearly on the latent state while still allowing efficient inference of filtered and smoothed beliefs. Thus, we introduce an auxiliary, intermediate representation $\mathbf{w}_t$ for the original observations $\mathbf{o}_t$. Such an intermediate representation allows us to capture the highly complex relations between state and observations in the mapping from $\mathbf{o}_t$ to $\mathbf{w}_t$ while using a simple observation model $q(\mathbf{w}_t|\mathbf{z}_t)$ in the latent space. This approach is common for variational latent State Space Models (Karl et al., 2016; Fraccaro et al., 2017; Klushyn et al., 2021) as it enables efficient inference in latent space. These approaches model $\mathbf{w}_t$ as a latent Gaussian random variable and learn it's mean and variance based on $\mathbf{o}_t$. They then sample $\mathbf{w}_t$ given these parameters. Yet, this assumption complicates inference and training, as now an additional latent variable, $\mathbf{w}_t$, has to be addressed. Further, it makes uncertainty propagation from observations to states harder (Becker et al., 2019; Volpp et al., 2021), as explicit uncertainty information is lost during sampling. Thus, following (Haarnoja et al., 2016; Becker et al., 2019; Shaj et al., 2020; Volpp et al., 2021), we instead formulate the observation model as $q(\mathbf{w}_t|\mathbf{z}_t) = \mathcal{N}\left(\mathbf{w}_t|\mathbf{z}_t, \boldsymbol{\sigma}_t^{\mathrm{w}}\right)$, where $\mathbf{w}_t = \text{enc}_o(\mathbf{o}_t)$ and the diagonal covariance $\boldsymbol{\sigma}^{\mathbf{w}_t} = \text{enc}_\sigma(\mathbf{o}_t)$ are given by an encoder network taking the original observation. In practice, the encoder consists of a single neural network with two output heads. This formulation does not suffer from the previously mentioned issues. First, as it models $\mathbf{w}_t$ as a direct mapping of the observation and not a random variable, we do not need to infer $\mathbf{w}_t$ but can assume it is observable. This assumption results in a simplified inference and training objective as we do not have to account for unobserved latent variables other than the latent state itself. Second, the encoder can directly extract the uncertainty in the observation, which tells the inference procedure how much it can rely on $\mathbf{w}_t$. For example, when estimating $\mathbf{w}_t$ from images, some images might contain certain information, e.g., the positions of an object, while others do not. The former case would result in a low uncertainty and the latter in a high one. Given $\mathbf{w_t}$ and $\boldsymbol{\sigma}_t^{\mathrm{w}}$ we can update belief states using the Kalman update, i.e., Bayes rule for Gaussian distributions,

$$q(\mathbf{z}_t|\mathbf{o}_{\leq t}, \mathbf{a}_{\leq t-1}) = \frac{1}{Z} q(\mathrm{enc}_o(\mathbf{o}_t)|\mathbf{z}_t) q(\mathbf{z}_t|\mathbf{o}_{\leq t-1}, \mathbf{a}_{\leq t-1}).$$

Here, $\boldsymbol{\sigma}_t^{\mathrm{w}}$ contributes to the computation of the Kalman gain, which highlights how the model uses the encoder's uncertainty to trade-off information from the prior belief and observation.

The inference observation model is combined with the shared locally linear dynamics model $q(\mathbf{z}_{t+1}|\mathbf{z}_t, \mathbf{a}_t) = p(\mathbf{z}_{t+1}|\mathbf{z}_t, \mathbf{a}_t)$ to obtain a latent LGSSM for inference. Given this dynamics model, we can forward beliefs in time using closed-form Gaussian marginalization,

$$q(\mathbf{z}_{t+1}|\mathbf{o}_{\leq t}, \mathbf{a}_{\leq t}) = \int q(\mathbf{z}_{t+1}|\mathbf{z}_t, \mathbf{a}_t) q(\mathbf{z}_t|\mathbf{o}_{\leq t}, \mathbf{a}_{\leq t-1}) d\mathbf{z}_t.$$

Repeating the described forms of forwarding beliefs in time and updating them based on observations amounts to standard (extended) Kalman filtering (Kalman, 1960; Jazwinski, 1970). Similarly, given these models, we can compute smoothed belief states using the Rauch-Tung-Striebel (Rauch et al., 1965) equations. In particular, the smoothing only requires beliefs and transition models computed during filtering and does not introduce additional learnable parameters. Note that both the filtering and smoothing equations simplify under our assumptions of diagonal covariances and are thus efficient and scalable. Thus, the smoothing inference only slightly increases the computational load during training. Especially in the image-based settings considered here, the computational cost largely stems from encoding and reconstructing observations.

5.3.3. Generative Model

Recall that we assume a generative State Space Model (SSM) consisting of three parts. Those are the dynamics model, the observation model, and the initial state distribution. First, we have the dynamics model $p(\mathbf{z}_{t+1}|\mathbf{z}_t, \mathbf{a}_t)$ as described in Section 5.3.1. Second, we assume a Gaussian generative observation model $p(\mathbf{o}_t|\mathbf{z}_t)$ parameterized by a neural network (decoder). Following Hafner et al. (2019), we parameterize the mean by a neural network and assume a fixed variance. This parameterization is often empirically beneficial, especially in the image-based setting considered in our experiments, but not a theoretical constraint. We could also parameterize the variance by the network if required. Third, we have an initial state distribution $p(\mathbf{z}_0)$. Here we use a Gaussian with zero mean and a learned diagonal variance which we initialize with the identity matrix.

5.3.4. Stochastic Gradient Variational Bayes for Model Learning

We train our model using a version of stochastic gradient variational Bayes (Kingma and Welling, 2013) and maximize $\mathcal{L}_{\mathrm{ssm}}$, (Equation 5.4). Like most approaches building on (Kingma and Welling, 2013) we approximate all expectations in Equation 5.4 using Monte Carlo estimation with a single sample from the latent variable and jointly optimize the parameters of q and p. For training, we infer the required belief states using the Kalman-based smoothing procedure described in Section 5.3.2. To compute the KL term in Equation 5.4 we additionally need the smoothed dynamics $q(\mathbf{z}_{t+1}|\mathbf{z}_t, \mathbf{a}_{\geq t}, \mathbf{o}_{\geq t+1})$ which we obtain with minimal overhead by extending the RTS equations as detailed in Appendix C.1.2.

The VRKN's formulation tightly couples posterior and smoothed beliefs by a fixed (not learned), deterministic, and well-motivated procedure. Smoothing using the Rauch-Tung-Striebel equations ensures that reasonable smoothed beliefs can only stem from reasonable posterior beliefs. Thus, while the posterior belief is not explicitly part of the objective, it is still learned during training as a prerequisite of the smoothed beliefs.

5.3.5. Using the Model for Online Control and Reinforcement Learning

When using the model for online control, we cannot smooth but have to act based on samples from the posterior belief $q(\mathbf{z}_t|\mathbf{a}_{\leq t}, \mathbf{o}_{\leq t})$. We can rely on Kalman filtering, as described in Section 5.3.2, to obtain this posterior belief and omit the smoothing step, as future observations are unavailable. For the VRKN, the tight coupling between smoothed and posterior beliefs ensures the posteriors are reasonable and usable for control.

To act based on the latent state beliefs, we add a decoder network to predict rewards from latent states. Following (Hafner et al., 2019), this decoder is trained by adding a reconstruction loss term to the objective. Given the predicted reward, various forms of control are applicable to act optimally. Yet, in this work, we focus on the underlying SSM and thus reuse two previously introduced methods building on the RSSM (Hafner et al., 2019, 2020). The PlaNet (Hafner et al., 2019) approach plans actions using the cross entropy method by rolling out trajectories on the model. The Dreamer (Hafner et al., 2020) approach learns a parametric policy and value function based on latent imagination. Such latent imagination uses the model as a differentiable simulator and optimizes the policy based on the estimated values of predicted future states.

5.3.6. Sensor Fusion

Given the possibility of using the Kalman update for incorporating observations, we can use the VRKN for a simple but principled approach to sensor fusion. Formally, we assume the observation $\mathbf{o}_t$ factorizes into K different observations $\mathbf{o}_t^{(k)}$, i.e., $p(\mathbf{o}_t|\mathbf{z}_t) = \prod_{k=1}^K p(\mathbf{o}_t^{(k)}|\mathbf{z}_t)$. Those observations can be of various modalities and be available at different frequencies, e.g., high-frequency velocity information from an inertial measurement unit and low-frequency camera images of the surroundings. In this scenario, we have K encoders and K decoders, one for each $\mathbf{o}_t^{(k)}$. Similar to more traditional sensor fusion approaches (Gustafsson, 2010), we then accumulate the intermediate observation representation $\mathbf{w}_t^{(k)}$ by repeatedly applying the Kalman update. This approach reflects the invariance to permutations of all observations for a single time step. Furthermore, it enables the model to omit the update if some of the K observations are unavailable for a time step.

5.4. Evaluation

A reference implementation of the Variational Recurrent Kalman Network with PlaNet and Dreamer Agents, as well as all baselines, is available at:

```
https://github.com/pbecker93/vrkn
```

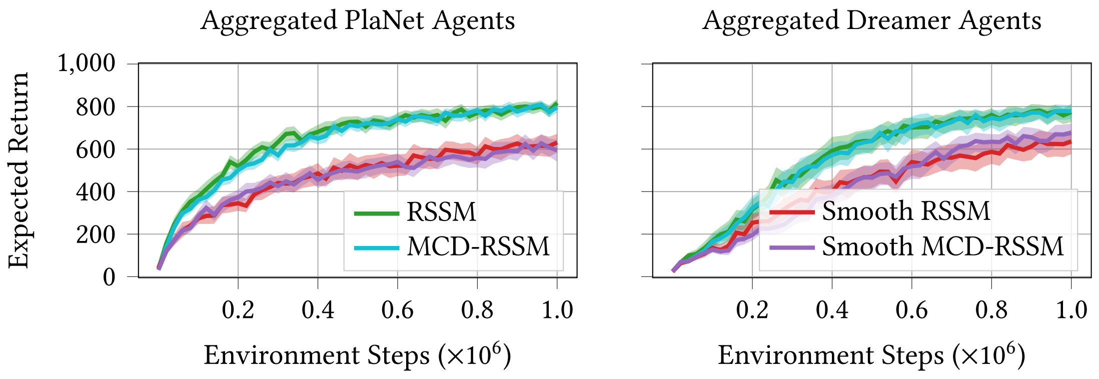


Figure 5.4.: Comparison of the RSSM, the simple smoothing extension (Smooth RSSM) as well as versions of both models using Monte Carlo Dropout (MCD) to capture epistemic uncertainty in the dynamics (MCD-RSSM and Smooth MCD-RSSM). Note that we use different tasks for PlaNet and Dreamer-based agents. Thus the left and right plots are not directly comparable. We find that proper inference by smoothing for both types of agents deteriorates performance. The additional epistemic uncertainty does not compensate for this decrease in performance.

We compare the VRKN, the original RSSM,and the modified RSSM version introduced in Section 5.2.3 on image-based continuous control tasks using the DeepMind Control Suite (Tassa et al., 2020). Prior works (Lambert et al., 2020; Lutter et al., 2021) concluded that the model's predictive performance is often uninformative about the quality of the model-based agent. We concluded the same after preliminary experiments and want to study the effects of the different assumptions and parametrizations on the performance in a model-based RL setting. Thus, we evaluate the SSMs as backbones for model-based agents and directly consider the achieved reward. As mentioned, we use both the PlaNet and the Dreamer approaches for control and thus closely follow the experiment setup described by Hafner et al. (2019, 2020). Appendix C.2 details the experimental setup and used baselines.

For our experiments, we run a varying number of tasks from the DeepMind Control Suite (Tassa et al., 2020) and report aggregated results over these tasks. Such aggregation is possible as the rewards in the DeepMind Control Suite are normalized, and all sequences are of equal length (1, 000 steps). Thus the returns are normalized, and aggregation yields better performance estimates (Agarwal et al., 2021). Again following the suggestions of Agarwal et al. (2021), we report interquartile means while indicating 95% stratified bootstrap confidence intervals by shaded areas. These metrics are computed using the provided library[2]. We base our conclusions on those aggregated results. Unless noted otherwise, we use 10 seeds for each agent-environment pair, train for 1 million environment steps, and approximate the expected return using 10 rollouts. Appendix C.3 provides reward curves for the individual tasks and additional quantitative results, such as box plots of their final performance.

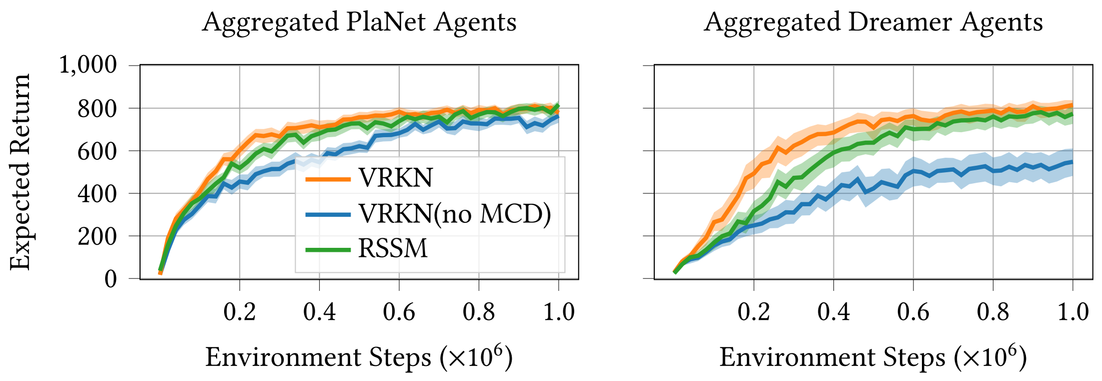


Figure 5.5.: Comparison of VRKN and RSSM-based PlaNet and Dreamer agents on standard DeepMind Control Suite tasks. Note that we use different tasks for PlaNet and Dreamer-based agents. Thus the left and right plots are not directly comparable. The VRKN without epistemic uncertainty (VRKN (no MCD)) cannot compete with the RSSM-agents, especially using the more involved Dreamer-based agents. Yet, the VRKN profits from the epistemic uncertainty, and closes the performance gap to the RSSM and even shows a tendency to be more sample efficient.

5.4.1. Evaluation of the Effect of Epistemic Uncertainty on Different Smoothing Architectures

We start our evaluation by comparing the original RSSM with its extended smoothing version using a GRU with and without Monte Carlo Dropout (MCD), described in Section 5.2.3. We also include a version of the original RSSM with MCD to model epistemic uncertainty for completeness. For the PlaNet agents, we evaluate the 6 tasks originally used in (Hafner et al., 2019). For the Dreamer agents, we use 8 tasks. Those are `Cheetah Run`, `Walker Walk`, `Cartpole Swingup`, `Cup Catch`, `Reacher Easy`, `Hopper Hop`, `Pendulum Swingup`, and `Walker Run`. As indicated by the results in Figure 5.4, the proper smoothing inference significantly deteriorates performance. Adding epistemic uncertainty in the form of MCD does, on average, neither affect the performance of the original RSSM nor the smoothing RSSM. These results underpin the argument that naive smoothing with the RSSM gives suboptimal performance. Additionally, they show that epistemic uncertainty in the form of Monte Carlo Dropout cannot serve as a remedy for the RSSM.

Next, we compare the VRKN to the RSSM using the same environments and also include a version of the VRKN without Monte Carlo Dropout (MCD), i.e., without epistemic uncertainty, dubbed VRKN (no MCD). Figure 5.5 shows the aggregated results for both PlaNet and Dreamer agents. Those results demonstrate that matching the RSSM's performance is possible with a principled smoothing inference by explicitly modeling epistemic uncertainty and providing a more appropriate inductive bias. Additionally, we find the VRKN tends to reach a given performance with fewer environment interactions, especially for the Dreamer agents.

Together, the results from Figure 5.4 and Figure 5.5 emphasize the importance of regularization for model-based RL. This regularization can be either implicitly by suboptimal inference or explicitly by capturing epistemic uncertainty. Additionally, they indicate that epistemic uncertainty alone is insufficient for approaches using a correct smoothing inference. Yet, the VRKN's appropriate

[2] `rliable`: `https://github.com/google-research/rliable`

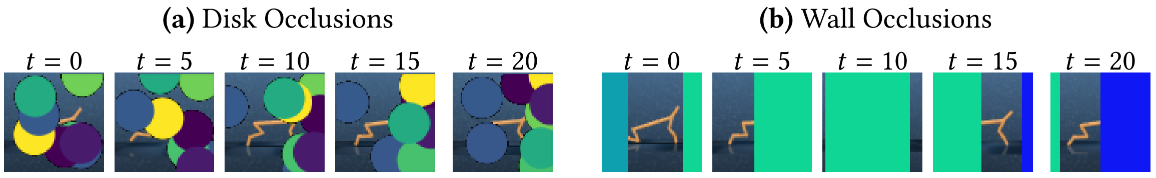


Figure 5.6.: We introduce two types of occlusions to induce partial observability. **(a)** Disk Occlusion: Slow-moving disks float through the image and bounce off its walls. **(b)** Wall Occlusions: Walls slide over the image from right to left at a constant speed. Their width is sampled randomly between half the image width and the image width. In both cases, the models have to accurately estimate which parts of the system are visible and which are not. Yet, the wall-based occlusions are more correlated over time and require memorization and prediction over longer time periods.

inductive bias makes the more principled approach of combining proper inference with explicit epistemic uncertainty feasible.

5.4.2. Evaluation on Tasks where Aleatoric Uncertainty Matters

The standard versions of the Deep Mind Control Suite Tasks have a deterministic simulation and rendering process. Thus, their aleatoric uncertainty is low. To better analyze the approaches' capabilities to capture and handle aleatoric uncertainty, we design tasks where doing so is necessary. To this end, we modify the `Cheetah Run`, `Walker Walk`, `Cartpole Swingup`, and `Cup Catch` tasks. First, we introduce transition noise by adding Gaussian noise to the actions before execution. We added noise with a standard deviation of 0.2 for `Cheetah Run` and `Walker Walk` and 0.3 for `Cartpole Swingup` and `Cup Catch`. Note that valid actions are between -1 and 1 for all tasks. Second, we modify the observations to contain only partial information, are missing for several time steps, or are available in different modalities at different frequencies. For details, we refer to the individual experiments below.

We use these tasks to study the effects of appropriately capturing aleatoric uncertainty in tasks where this form of uncertainty matters. We compare VRKN-based and the RSSM-based Dreamer agents and find that precise estimation of the aleatoric uncertainty makes a difference. Contrary to the original tasks, where RSSM and VRKN-based agents performed similarly, agents building on the VRKN outperform their RSSM-based counterparts in all considered scenarios.

5.4.2.1. Partial Observability through Occlusions

In a first approach to include observation uncertainty, we render two types of occlusions over the images, i.e., discs and walls. See Figure 5.6 for an explanation and some examples. Due to these occlusions, the individual observations have varying amounts of relevant information. Thus, the models need to correctly capture uncertainties in the system, allowing them to trade off information from the prior belief and current observation. For training, we use masked reconstruction, i.e., only non-occluded pixels contribute to the reconstruction loss. We want to emphasize that we do not consider the availability of such loss masks a realistic scenario but see the task as a reasonable benchmark to evaluate the models' capabilities to cope with uncertainties. We also consider a baseline where we train the model solely to reconstruct the reward to show that the approaches can still extract information from the highly occluded observations. Figure 5.7 shows the results of

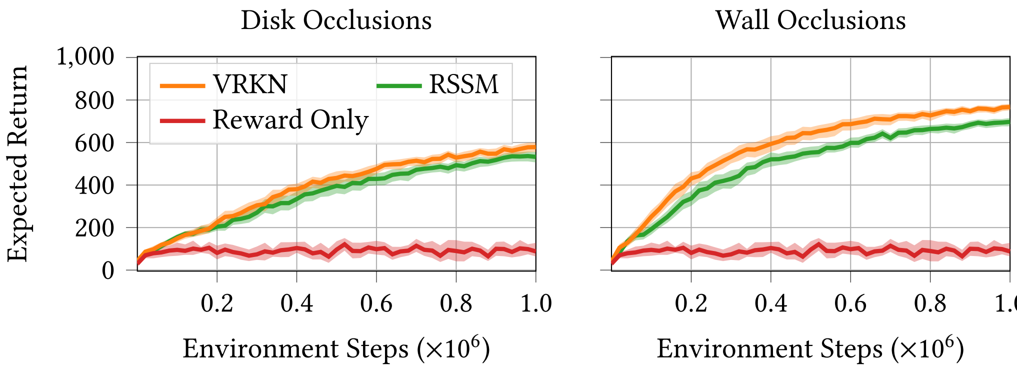


Figure 5.7.: Aggregated results for the partial observability experiments with both occlusion types. For the experiments, except for the reward-only baseline, we used 20 seeds per task. Both approaches clearly perform better than the reward-only baseline, showing that they can extract useful information from the occluded images. Yet, especially in the wall-occlusion task, the VRKN-based agents outperform the RSSM-based agents. They do not only achieve a higher reward but also converge faster. Especially in the wall occlusion task, it is insufficient to only extract information from the occluded images, but a reasonable belief also needs to be sustained over time.

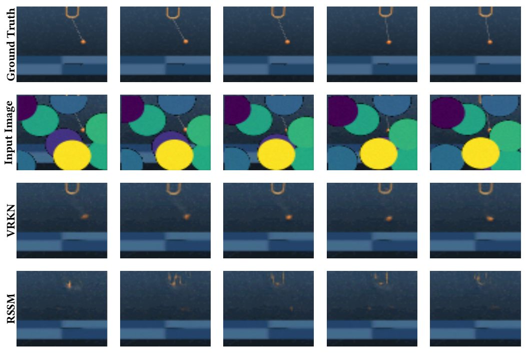


Figure 5.8.: Exemplary sub-sequence of reconstructions, based on the model's posterior beliefs. The first row is the noise-free ground truth image, which the models never see. The second row is the model input, followed by the VRKN and the RSSM reconstructions. Even though the ball is partly visible in most images, the RSSM fails to reconstruct its position. The VRKN manages to do so and even provides a reasonable estimate for cup position. These results indicate the VRKN's improved ability to capture the system state in noisy scenarios.

the comparison. Under the heavy occlusions considered here, the VRKN performs significantly better than the RSSM. In particular for the wall occlusions, the VRKN-based agents achieve a higher reward using fewer environment interactions. The wall occlusions are more correlated, i.e., occluded parts in one image are more likely also occluded in the previous and next images. Thus, they test the approaches' capability to not only form a reasonable belief but also to propagate it for multiple time steps.

Additionally, we qualitatively compare images reconstructed from posterior beliefs of both approaches to gain further insights into the quality of the belief state. Figure 5.8 shows some of those reconstructions for the `Cup Catch` task, further images can be found in Appendix C.3. From these images, it appears that the VRKN better captures the actual system state and uncertainty and thus allows the model-based agent to achieve a higher reward. Yet, while the reconstructions are visually much better, we only observe a small increase in performance, which we attribute to objective mismatch.

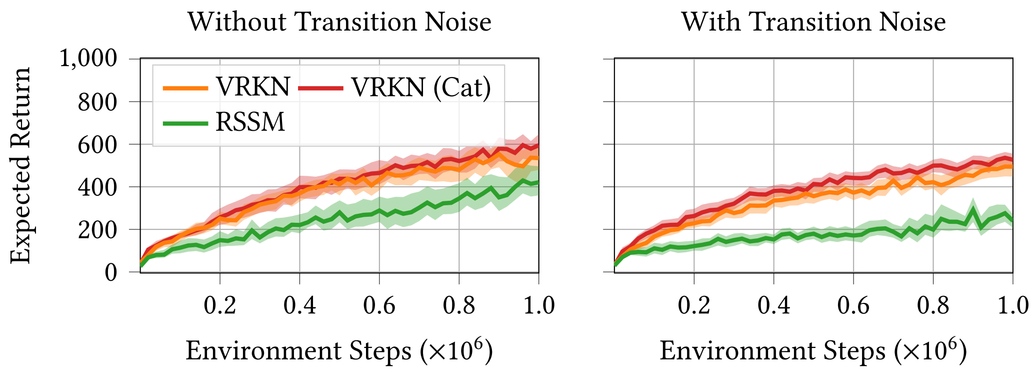


Figure 5.9.: Aggregated results in the missing observations setting with and without transition noise. First, we see that the VRKN works equally well if we provide the information about which information are valid explicitly (VRKN) or if we provide it by concatenation (VRKN (Cat)). Second, the RSSM performs significantly worse than both VRKN versions. Especially if the system is subject to transition noise it fails to give good results.

5.4.2.2. Dealing with Missing Observations

Another common setting where estimating aleatoric uncertainty is important is missing observations. It requires the models to propagate a reasonable belief without information and update it if an observation is available. Additionally, if no observations are available for multiple time steps, the belief state gradually drifts away from the real state due to noise and model inaccuracies. Once a new observation arrives, the discrepancy between this observation and the belief has to be explained. Recall that, due to its inference assumptions, the RSSM must explain the discrepancy by the last transition and thus will overestimate the aleatoric uncertainty even further. On the other hand, the VRKN with its smoothing inference can capture the drift of the belief state and evenly attribute the discrepancy to all transitions since the last observation. Thus the VRKN should be better equipped to solve such a task.

We consider a setting where we only provide every n-th image, where n is sampled uniformly between 4 and 8. We assume knowledge about whether the observation for a given time step is valid and feed that knowledge to the models as an additional input flag. For valid observation, the model receives the observation and a 1. For invalid observations, it receives a default value and a 0. The RSSM handles this flag by appending it to the convolution network processing the inputs or to the default value. See Appendix C.2.3 for details. The VRKN allows for two ways of handling the flag. First, we can again concatenate it to the observation before computing the latent observation and corresponding uncertainty estimate (VRKN (Cat)). Second, the VRKNs design allows us to provide the information more explicitly and the Kalman update step for invalid observations during inference. We evaluate with and without additional transition noise and show the results in Figure 5.9. Those results underpin the initial hypotheses that the VRKN is better equipped to learn in this setting and significantly outperforms the RSSM. The improvement is independent of how we provide information about missing information to the VRKN and, as expected, more significant with transition noise.

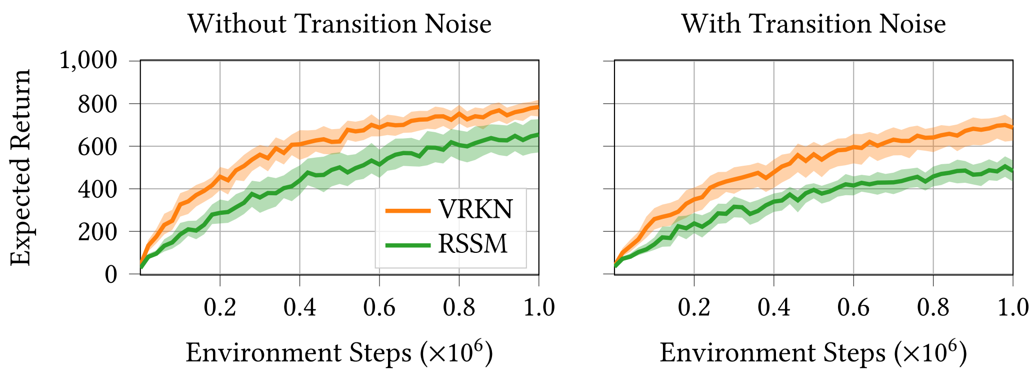


Figure 5.10.: Aggregated results in the fusion setting with and without transition noise. When comparing to Figure 5.9 we see that both approaches are able to exploit the additional proprioceptive information and improve their performance. Yet, the VRKN seems to better exploit and accumulate the available information, as the resulting agents significantly outperform their RSSM-based counterparts.

5.4.2.3. Fusing Information from Multiple Sensors at Different Frequencies

We extend the missing observations setting using proprioceptive information which is available at every time step. The exact form of the proprioceptive information is task-dependent. For example, for `Cup Catch`, we define the cup position as proprioceptive, but not the ball position, which has to be inferred from images. Appendix C.2.4 provides an overview for all considered tasks. This experiment mimics a common robotics scenario where we have proprioceptive information about the robot at high frequencies but need to estimate the environment's state based on lower-frequency images. It tests the approaches' abilities to form reasonable state estimates from observations that arrive in different modalities and at varying frequencies. Here, the models need to trade off information encoded in the prior belief with the information available in both sensor sources. This trade-off requires accurate estimates about the aleatoric uncertainty in both state and observations. For the fusion task, the VRKN can rely on the natural fusion mechanism described in Section 5.3. For the RSSM, we also use multiple encoder networks, decoders, and loss terms and concatenate the outputs of the encoders before forming the posterior belief. For details, see Appendix C.2.3.

As shown in Figure 5.10, the VRKN achieves a higher reward than the RSSM, especially in the setting with transition noise. This result again emphasizes the VRKNs capability to exploit all available information, appropriately capture the system's uncertainty, and form belief states that yield good performance.

5.5. Related Work

Recurrent State Space Models. While earlier works (Wahlström et al., 2015; Watter et al., 2015; Banijamali et al., 2018; Ha and Schmidhuber, 2018; Buesing et al., 2018) discussed and showed the feasibility of control using learned latent State Space Models, the work originally proposing the RSSM (Hafner et al., 2019) was the first to show that such approaches can achieve similar performance to model-free RL on pixel-based complex continuous control tasks while using significantly fewer environment interactions. Since then, Hafner et al. (2020) improved their approach using a parametric policy learned on imagined trajectories and categorical latent spaces

(Hafner et al., 2021). These approaches gained interest in the model-based RL community and are empirically successful, yet, little attention has been paid to the underlying SSM itself, the assumptions it builds upon, and its parametrization.

State Space Models. The Machine Learning community extensively studied and used SSMs. Besides classical approaches using linear models (Shumway and Stoffer, 1982) and works using Gaussian Processes (Eleftheriadis et al., 2017; Doerr et al., 2018), most recent methods build on Neural Networks (NNs). The first class of NN-based models of particular relevance for this work embeds Linear Gaussian SSMs (LGSSM) into latent spaces (Watter et al., 2015; Karl et al., 2016; Fraccaro et al., 2017; Banijamali et al., 2018; Becker-Ehmck et al., 2019; Klushyn et al., 2021). These approaches assume actuated systems and learn using stochastic gradient variational Bayes (Kingma and Welling, 2013). Yet, non of these approaches were used to model or even control systems of the complexity considered by (Hafner et al., 2019) and here. They are not directly applicable to these scenarios for various reasons. First, they use full transition matrices and covariances, which prevents them from scaling to sufficiently high-dimensional latent spaces. (Karl et al., 2016; Becker-Ehmck et al., 2019) do not allow smoothing. (Fraccaro et al., 2017; Klushyn et al., 2021) model the latent observations as random variables which are inferred jointly with the latent states and use constant observation uncertainty for the filtering in the latent space. This choice complicates inference and training and prevents principled usage of the observation uncertainty for filtering. Our parameterization of the LGSSM alleviates these issues by building on factorization assumptions which yield a scalable architecture. Further, it allows smoothing and principled usage of observation uncertainty during filtering by modeling the observations in latent space as deterministic. Finally, non of these approaches considered modeling epistemic uncertainty.

Another class of approaches directly uses NN-based, nonlinear parametrization for SSMs (Archer et al., 2015; Krishnan et al., 2015; Gu et al., 2015; Zheng et al., 2017; Krishnan et al., 2017; Yingzhen and Mandt, 2018; Schmidt and Hofmann, 2018; Naesseth et al., 2018; Moretti et al., 2019). Out of this class, Structured Inference Networks (SINs) (Krishnan et al., 2017) are the most relevant for our work. SINs build on the same variational objective as VRKN, yet without conditioning on actions. The smoothing-RSSM baseline introduced in Section 5.2.3 can be considered an instance of a SIN. Yet while it builds on the same loss and fundamental ideas, the underlying NN architecture is different.

Kalman Updates in Deep Latent Space. Haarnoja et al. (2016) first proposed using an encoder to extract uncertainty estimates from high-dimensional observations for filtering. They only learned the encoder while assuming the transition dynamics to be known. Becker et al. (2019) proposed an efficient factorization to additionally learn a high-dimensional, latent, locally-linear dynamics model. Shaj et al. (2020) extended this approach by introducing a principled form of action conditioning. While the VRKN builds on many of their design choices, there are also considerable differences. Those differences mainly concern the parametrization of the dynamics model and further simplifying the factorization assumptions. These changes are necessary to make the approaches scale to the complex control tasks considered in this work. Additionally, Haarnoja et al. (2016); Becker et al. (2019); Shaj et al. (2020) train using regression and do not learn a full generative model. Thus, they cannot produce the reasonable latent trajectories needed for model-based RL.

Epistemic Uncertainty for Model-Based RL. Ample work emphasizes the importance of modeling epistemic uncertainty for model-based RL (Deisenroth and Rasmussen, 2011; Chua et al., 2018; Janner et al., 2019) and several authors equipped RSSMs with epistemic uncertainty. Okada et al.

(2020) use an ensemble of RSSMs and showed improved results on modified versions of the Deep Mind Control Suite (Tassa et al., 2020) benchmarks. Sekar et al. (2020) also combine an ensemble with the RSSM but focus on exploration and generalization to unseen tasks. Yet, neither of these works questioned the assumptions underlying the RSSM or analyzed their effects on the learned models.

5.6. Conclusion

We analyzed the independence assumptions underlying Recurrent State Space Models (RSSMs) and found they are theoretically suboptimal. Yet, they implicitly regularize the model by causing an overestimated aleatoric uncertainty and are crucial to the RSSMs success in model-based RL. When trying to avoid this heuristic approach and use the correct assumptions while replacing the implicit regularization with a more explicit approach using epistemic uncertainty, we found a simple extension of the RSSM architecture is insufficient. Thus, we redesigned the model using well-understood and established components, providing a more appropriate inductive bias for smoothing. The resulting Variational Recurrent Kalman Network (VRKN) uses a latent Linear Gaussian State Space Model to address aleatoric uncertainty and Monte-Carlo Dropout to model epistemic uncertainty explicitly. Building on an LGSSM allows exact inference in the latent space using Kalman filtering and smoothing to obtain both smoothed and posterior belief states efficiently. While agents based on the VRKN and the RSSM perform similarly on the standard DeepMind Control Suite (Tassa et al., 2020) benchmarks, the VRKN-based agents significantly outperform those using the RSSM on tasks where capturing uncertainties is more relevant. Additionally, the VRKN provides a natural approach to sensor fusion and outperforms the RSSM on tasks that require fusing sensor observations from several sensors at different frequencies.

Limitations. We showed that designing a State Space Model out of well-founded components that matches or improves the RSSM's performance is possible. This insight opens a path to improve them individually. Yet, here we used simple instances of these components and have not yet further investigated how to improve them. Further, we have not investigated the interplay between the models and the controllers used on top of them but used the control approaches proposed in (Hafner et al., 2019) and (Hafner et al., 2020) with default parameters. Due to the intricate interplay between model learning and using the resulting controller for data collection, it is reasonable to rethink the design of the controller when changing the model.

Contribution Statement. *I conducted the analysis of the RSSM inference assumptions and implication for model learning, devised, derived, and implemented the VRKN, devised, implemented, and ran all the evaluations and wrote the large majority of the article. GN advised me during the project, assisted with the analysis and suggested experiments, and helped with writing the article.*

6. KalMamba: Towards Efficient Probabilistic State Space Models for RL under Uncertainty

While the Variational Recurrent Kalman Network (VRKN) introduced in Chapter 5 provides a principled way to model and disentangle epistemic and aleatoric uncertainties in world models, it is still computationally expensive. Its inherently sequential nature limits the parallelism of the model, making it challenging to scale. This chapter addresses this limitation and, together with the VRKN itself, answers **Q2**:

Q2 How can we build a computationally efficient world model for RL that captures and propagates uncertainties in the world while separating them from uncertainties due to lack of training data?

Our approach, KalMamba, bridges the gap between the VRKN and a concurrently emerging class of deterministic structured SSMs (Gu et al., 2021; Smith et al., 2022; Gu and Dao, 2023) that can process sequential data in parallel. While these consider different approaches for efficient computation, the most intuitive and flexible ideas build on parallel scan algorithms (Blelloch, 1990; Harris et al., 2007), which allow parallelized sequential computations over associative binary operators. Crucially, Särkkä and García-Fernández (2020) (Section 2.1.3.2) showed that such parallel scan algorithms can efficiently compute the forward and backward passes of Kalman filters, creating a natural connection to the RKN and VRKN.

The following was published as KalMamba: Towards Efficient Probabilistic State Space Models for RL under Uncertainty *(Philipp Becker, Niklas Freymuth, and Gerhard Neumann, 2024a) at the Workshop on Aligning Reinforcement Learning Experimentalists and Theorist (ARLET 2024) at ICML 2024. Reprinted with permission of the authors. Wording, notation, and formulations were revised in several places. Shorter versions of this article were also presented at the Next Generation of Sequence Modeling Architectures Workshop at ICML 2024 and the Training Agents with Foundation Models Workshop at RLC 2024.*

6.1. Introduction

Deep probabilistic State Space Models (SSMs) are versatile tools widely used in Reinforcement Learning (RL) for environments with high-dimensional, partial, or noisy observations (Hafner et al., 2020; Lee et al., 2020; Nguyen et al., 2021; Becker and Neumann, 2022; Hafner et al., 2023; Samsami et al., 2024). They model states and observations as random variables and relate them through a set of conditional distributions, allowing them to capture uncertainties and learn concise

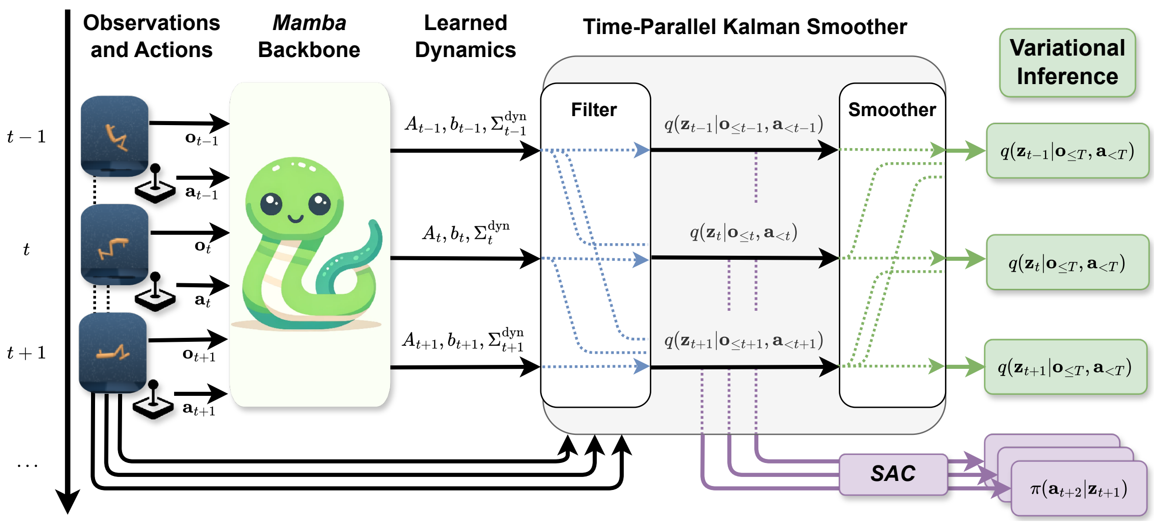

Observations and Actions
Mamba Backbone
Learned Dynamics
Time-Parallel Kalman Smoother
Variational Inference
Filter
Smoother
$t-1$
t
$t+1$
$\dots$
$\mathbf{o}_{t-1}$
$\mathbf{a}_{t-1}$
$\mathbf{o}_t$
$\mathbf{a}_t$
$\mathbf{o}_{t+1}$
$\mathbf{a}_{t+1}$
$A_{t-1}, b_{t-1}, \Sigma_{t-1}^{\mathrm{dyn}}$
$A_t, b_t, \Sigma_t^{\mathrm{dyn}}$
$A_{t+1}, b_{t+1}, \Sigma_{t+1}^{\mathrm{dyn}}$
$q(\mathbf{z}_{t-1}|\mathbf{o}_{\leq t-1}, \mathbf{a}_{<t-1})$
$q(\mathbf{z}_t|\mathbf{o}_{\leq t}, \mathbf{a}_{<t})$
$q(\mathbf{z}_{t+1}|\mathbf{o}_{\leq t+1}, \mathbf{a}_{<t+1})$
$q(\mathbf{z}_{t-1}|\mathbf{o}_{\leq T}, \mathbf{a}_{<T})$
$q(\mathbf{z}_t|\mathbf{o}_{\leq T}, \mathbf{a}_{<T})$
$q(\mathbf{z}_{t+1}|\mathbf{o}_{\leq T}, \mathbf{a}_{<T})$
SAC
$\pi(\mathbf{a}_{t+2}|\mathbf{z}_{t+1})$

Figure 6.1.: Overview of KalMamba. The observation-action sequences are first fed through a dynamics backbone built on Mamba (Gu and Dao, 2023) to learn a linear dynamics model for each step. KalMamba then uses time-parallel Kalman filtering (Särkkä and García-Fernández, 2020) to infer filtered beliefs $q(\mathbf{z}_t|\mathbf{o}_{\leq t}, \mathbf{a}_{\leq t-1})$ which can be used for control with a Soft Actor Critic (SAC) (Haarnoja et al., 2018). For model training, KalMamba employs an additional time-parallel Kalman smoothing step to obtain smoothed beliefs $q(\mathbf{z}_t|\mathbf{o}_{\leq T}, \mathbf{a}_{\leq T})$. These beliefs allow training a model that excels in modeling uncertainties due to a tight variational lower bound (Becker and Neumann, 2022). Crucially, the smoothing step does not introduce trainable model parameters, enabling the direct use of the filtered beliefs for downstream RL policy training and execution.

probabilistic representations for downstream RL applications. Beyond RL, recent deterministic SSMs (Gu et al., 2021; Smith et al., 2022; Gu and Dao, 2023) offer a powerful new paradigm for general sequence modeling and rival state-of-the-art transformers while improving computational complexity (Gu and Dao, 2023). These models assume states and observations are vectors related by deterministic, linear, and associative functions, which allow efficient time-parallel computations. Such deterministic models are often insufficient for RL with complex observations, where uncertainty awareness and probabilistic modeling are crucial (Chua et al., 2018; Lee et al., 2020; Hafner et al., 2019). In contrast, due to their nonlinear parameterizations and inference approaches, most existing probabilistic SSMs for RL and beyond do not feature the favorable scaling behavior of recent deterministic SSMs.

Many real-world applications require both uncertainty awareness and the capability of handling long sequences. Examples include multimodal robotics tasks with high-frequency control, long sequence non-stationary tasks, or complex information-gathering tasks. Consider a robot tasked with packing objects of unknown properties into a basket. By interacting with each item to infer and memorize properties such as mass and deformability, the robot refines its understanding of the scene, enabling it to strategically arrange the objects in the basket. Current deterministic SSMs lack uncertainty awareness to solve such tasks, while their probabilistic counterparts do not scale to the required sequence lengths. Thus, the question of how to develop a principled method that combines the benefits of both paradigms to obtain robust and efficient probabilistic SSMs for long-sequence RL under uncertainty arises.

We propose an efficient architecture for RL that equips probabilistic SSMs with the efficiency of recent deterministic SSMs. Our approach, KalMamba, uses (extended) Kalman filtering and smoothing (Kalman, 1960; Rauch et al., 1965; Jazwinski, 1970) to infer belief states over a Linear Gaussian SSM in a latent space that uses a dynamics model based on Mamba (Gu and Dao, 2023). In this approach, Mamba acts as a highly effective general-purpose sequence-to-sequence model to learn the parameters of a dynamics model. The Kalman Smoother uses this model to compute probabilistic beliefs over system states. Figure 6.1 provides a schematic overview. Mamba is efficient for long sequences as it uses parallel associative scans, which allow parallelizing associative operators on highly parallel hardware accelerators such as GPUs (Sengupta et al., 2007). Similarly, we formulate both Kalman filtering and smoothing as associative operations (Särkkä and García-Fernández, 2020) and build efficient parallel scans for filtering and smoothing in PyTorch (Paszke et al., 2019). With both Mamba and the Kalman Smoother being parallelizable, KalMamba achieves time-parallel computation of belief states required for model learning and control. Thus, unlike previous approaches for efficient SSM-based RL (Samsami et al., 2024), which rely on simplified inference assumptions, KalMamba enables end-to-end model training under high levels of uncertainty using a smoothing inference and tight variational lower bound (Becker and Neumann, 2022). While using smoothed beliefs for model learning, our architecture ensures a tight coupling between filtered and smoothed belief states. This inductive bias ensures the filtered beliefs are meaningful, allowing their use for policy learning and execution where future observations are unavailable.

We evaluate KalMamba on several tasks from the DeepMind Control (DMC) Suite (Tassa et al., 2018), training an off-the-shelf Soft Actor-Critic (Haarnoja et al., 2018) on beliefs inferred from both images and states. As baselines, we compare to Recurrent State Space Models (Hafner et al., 2019) and the Variational Recurrent Kalman Network (Becker and Neumann, 2022). Our preliminary experiments show that KalMamba is competitive to these state-of-the-art SSMs while being significantly faster to train and scaling gracefully to long sequences due to its ability to be efficiently parallelized. These results indicate KalMamba's potential for applications that require forming accurate belief states over long sequences under uncertainty.

To summarize our contributions, we **(i)** propose KalMamba, a novel probabilistic SSM for RL that combines Kalman filtering, smoothing, and a Mamba backbone to offer efficient probabilistic inference, **(ii)** motivate and compare KalMamba to existing probabilistic SSMs for RL, and **(iii)** validate our approach on state- and image-based control tasks, closely matching the performance of state-of-the-art probabilistic SSMs while being time-parallelizable.

6.2. Related Work

Deterministic State Space Models in Deep Learning. Structured deterministic State Space approaches (Gu et al., 2021; Smith et al., 2022; Gu and Dao, 2023) recently emerged as an alternative to the predominant Transformer (Vaswani et al., 2017) architecture for general sequence modeling (Gu and Dao, 2023). Their main benefit is combining compute and memory requirements that scale linearly in sequence length with efficient and parallelizable implementations. While earlier approaches, such as the Structured State Space Sequence Model (S4) (Gu et al., 2021) and others (Gupta et al., 2022; Hasani et al., 2022) used a convolutional formulation for efficiency, more recent approaches (Smith et al., 2022; Gu and Dao, 2023) use associative scans. Such associative scans allow for parallel computations over sequences if all involved operators are associative, which yields a logarithmic runtime, given enough parallel cores. However, all these models are deterministic, i.e., they do not model uncertainties or allow sampling without further modifications. As a remedy, Latent S4 (LS4) (Zhou et al., 2023) extends S4 for probabilistic generative sequence modeling and forecasting. However, in LS4, the latent states are not Markovian and are thus hard to use for control. KalMamba exploits the fact that filtering and smoothing in Linear Gaussian State Space Models (LGSSM) models can also be formulated as a set of associative operations, which makes it amenable to parallel scans (Särkkä and García-Fernández, 2020). To our knowledge, it is the first deep-learning model to do so. Further, it relies on Mamba (Gu and Dao, 2023), a state-of-the-art deterministic State Space Models, to precompute the dynamics models required for filtering and smoothing.

Probabilistic State Space Models for Reinforcement Learning. Probabilistic State Space Models (SSMs) are commonly and successfully used for Reinforcement Learning (RL) from high dimensional or multimodal observations (Nguyen et al., 2021; Wu et al., 2022; Hafner et al., 2023; Becker et al., 2024b), under partial observability (Becker and Neumann, 2022), and for memory tasks (Samsami et al., 2024). Arguably, the most prominent approach is the Recurrent State Space Model (RSSM) (Hafner et al., 2019). After their original introduction as the basis of a standard planner, they have been improved with more involved parametric policy learning approaches Hafner et al. (2020) and categorical latent variables for categorical domains (Hafner et al., 2021). During inference, the RSSMs conditions the latent state on past observations and actions, resulting in a filtering inference scheme. Here, the key architectural feature of RSSMs is splitting the latent state into stochastic and deterministic parts. The deterministic part is then propagated through time using a standard recurrent architecture. In its original formulation, the RSSM uses a Gated Recurrent Unit (GRU) (Cho et al., 2014). One line of research focuses on replacing this deterministic path with more efficient architectures with the TransDreamer (Chen et al., 2021) approach using a transformer (Vaswani et al., 2017) and Recall to Image (Samsami et al., 2024) using S4 (Gu et al., 2021). However, to fully exploit the efficiency of these backbone architectures, both need to simplify the inference assumptions and can only consider the current observation, which makes them highly susceptible to noise or missing observations. Opposed to that, the Variational Recurrent Kalman Network (VRKN) (Becker and Neumann, 2022) proposes using a smoothing inference scheme that conditions both past and future actions. This scheme allows the VRKN to work with a fully stochastic latent state and lets it excel in tasks where modeling uncertainty is crucial. The VRKN uses a locally LGSSM in a latent space, performing closed-form Kalman Filtering and smoothing. KalMamba holistically combines smoothing inference in a fully probabilistic SSM with an efficient temporally parallelized implementation, resulting in an approach that is robust to noise and efficient.

Probabilistic State Space Models in Deep Learning. Probabilistic SSMs are versatile and commonly used tools in machine learning. Besides classical approaches using linear models (Shumway and Stoffer, 1982) and works using Gaussian Processes (Eleftheriadis et al., 2017; Doerr et al., 2018), most recent methods build on Neural Networks (NNs) to parameterize generative and inference models using the SSM assumptions (Archer et al., 2015; Watter et al., 2015; Gu et al., 2015; Karl et al., 2016; Fraccaro et al., 2017; Krishnan et al., 2017; Banijamali et al., 2018; Yingzhen and Mandt, 2018; Schmidt and Hofmann, 2018; Naesseth et al., 2018; Becker et al., 2019; Becker-Ehmck et al., 2019; Moretti et al., 2019; Shaj et al., 2020; Klushyn et al., 2021; Shaj et al., 2022). Out of these approaches, those that embed Linear Gaussian SSMs into latent spaces (Watter et al., 2015; Haarnoja et al., 2016; Fraccaro et al., 2017; Banijamali et al., 2018; Becker-Ehmck et al., 2019; Becker et al., 2019; Shaj et al., 2020; Klushyn et al., 2021; Shaj et al., 2022) are of particular relevance to KalMamba. Doing so allows for closed-form inference using (extended) Kalman Filtering and

Method	Inference Model	Smooth	Parallel
RSSM (Hafner et al., 2019)	$q(\mathbf{z}_t \mid \mathbf{h}_t, \mathbf{o}_t)$	×	×
R2I (Samsami et al., 2024)	$q(\mathbf{z}_t \mid \mathbf{o}_t)$	×	✓
VRKN (Becker and Neumann, 2022)	$q(\mathbf{z}_t \mid \mathbf{o}_{\leq T}, \mathbf{a}_{\leq T})$	✓	×
KalMamba	$q(\mathbf{z}_t \mid \mathbf{o}_{\leq T}, \mathbf{a}_{\leq T})$	✓	✓

Table 6.1.: Comparing the inference models and capabilities for smoothing (Smooth) and time-parallel (Parallel) execution of recent SSMs for RL.

Smoothing. However, with the notable exception of the VRKN, these models usually cannot be used to control or even model systems of similar complexity to those controlled with RSSM-based approaches. Furthermore, some of them (Karl et al., 2016; Becker-Ehmck et al., 2019) do not allow smoothing, while others (Fraccaro et al., 2017; Klushyn et al., 2021) model observations in the latent space as additional random variables which complicates inference and training and prevents principled usage of the observation uncertainty for filtering. Another class of approaches (Haarnoja et al., 2016; Becker et al., 2019; Shaj et al., 2020, 2022) trains using regression and are thus not generative. Notably, none of these approaches uses a temporally parallelized formulation of the filtering and smoothing operations.KalMamba takes inspiration from many of these approaches and partly follows the VRKN's design to enable RL complex systems. However, it combines those ideas with the efficiency of recent deterministic SSMs using an architecture that enables time-parallel computations.

6.3. State Space Models for Reinforcement Learning

In Reinforcement Learning (RL) under uncertainty and partial observability, State Space Models (SSMs) generally assume sequences of observations $\mathbf{o}_{\leq T} = (\mathbf{o}_t)_{t=0\cdots T}$ which are generated by a sequence of latent state variables $\mathbf{z}_{\leq T} = (\mathbf{z}_t)_{t=0\cdots T}$, given a sequence of actions $\mathbf{a}_{\leq T} = (\mathbf{a}_t)_{t=0\cdots T}$. The corresponding generative model factorizes according to the hidden Markov assumptions (Murphy, 2012), i.e., each observation $\mathbf{o}_t$ only depends on the current latent state $\mathbf{z}_t$ through an observation model $p(\mathbf{o}_t \mid \mathbf{z}_t)$, and each latent state $\mathbf{z}_t$ only depends on the previous state $\mathbf{z}_{t-1}$ and the action $\mathbf{a}_{t-1}$ through a dynamics model $p(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{a}_{t-1})$.

In order to learn the SSMs from data and use it for downstream RL, we need to infer latent belief states given observations and actions. Depending on the information provided for inference, we differentiate between the filtered belief $\mathbf{q}(\mathbf{z}_t \mid \mathbf{o}_{\leq t}, \mathbf{a}_{\leq t-1})$ and the smoothed belief $\mathbf{q}(\mathbf{z}_t \mid \mathbf{o}_{\leq T}, \mathbf{a}_{\leq T})$. The filtered belief conditions only on past information, while the smoothed belief also depends on future information. Computing these beliefs is intractable for models of reasonable complexity. Thus, we resort to an autoencoding variational Bayes approach that allows joint training of the generative and an approximate inference model using a lower bound objective (Kingma and Welling, 2013).

The Recurrent State Space Model (RSSM) (Hafner et al., 2019) assumes a nonlinear dynamics model, splitting the state $\mathbf{z}_t$ into a stochastic $\mathbf{s}_t$ and a deterministic part $\mathbf{h}_t$ which evolve according to $\mathbf{h}_t = f(\mathbf{h}_{t-1}, \mathbf{a}_{t-1}, \mathbf{s}_{t-1})$ and $\mathbf{s}_t \sim p(\mathbf{s}_t \mid \mathbf{h}_t)$. Here f is implemented using a Gated Recurrent Unit (GRU) (Cho et al., 2014). This results in a nonlinear, autoregressive process that cannot be parallelized over time. Further, RSSMs assume a filtering inference model $q(\mathbf{s}_t \mid \mathbf{h}_t, \mathbf{o}_t)$, where $\mathbf{h}_t$

accumulates all information from the past. The RSSM's inference scheme struggles with correctly estimating uncertainties as the resulting lower bound is not tight (Becker and Neumann, 2022). In tasks where such uncertainties are relevant, this lack of principled uncertainty estimation causes poor performance for downstream applications.

As a remedy, the Variational Recurrent Kalman Network (VRKN) (Becker and Neumann, 2022) builds on a Linear Gaussian SSM in a latent space which allows inferring smoothed belief states $\mathbf{q}(\mathbf{z}_t|\mathbf{o}_{\leq T}, \mathbf{a}_{\leq T})$ required for a tight bound. The VRKN removes the need for a deterministic path and improves performance under uncertainty. However, it linearizes the dynamics model around the mean of the filtered belief, resulting in a nonlinear autoregressive process that cannot be parallelized.

In contrast, Recall to Image (R2I) (Samsami et al., 2024) builds on the RSSM and improves computational efficiency at the cost of a more simplistic inference scheme. It uses S4 (Gu et al., 2021) instead of a GRU to parameterize the deterministic path f but has to remove the inference's dependency on $\mathbf{h}_t$ to allow efficient parallel computation. The resulting inference model, $q(\mathbf{z}_t|\mathbf{o}_t)$ is non-recurrent and neglects all information from other time steps. Thus, while R2I excels on memory tasks, it is highly susceptible to noise and partial-observability as the inference cannot account for inconsistent or missing information in $\mathbf{o}_t$.

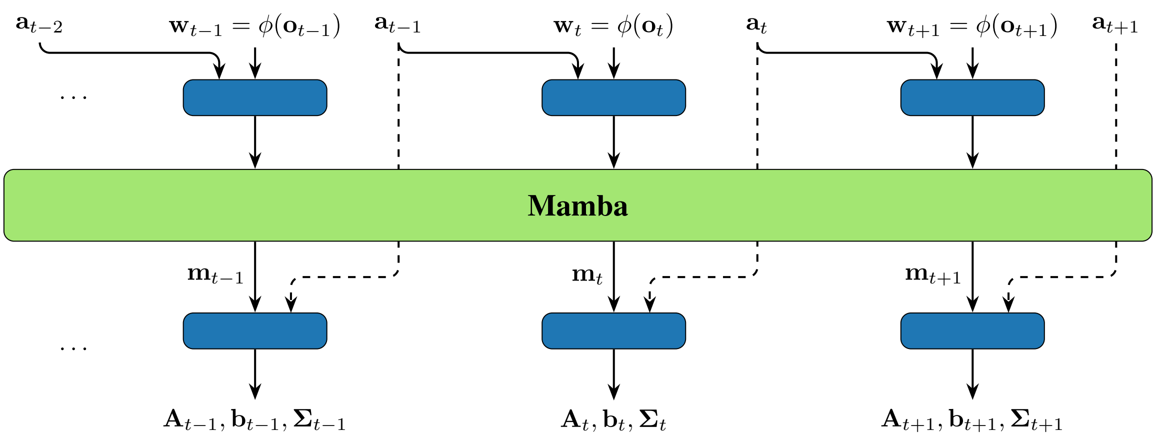
Our approach, KalMamba, combines the tight variational lower bound of the VRKN with a parallelizable Mamba (Gu and Dao, 2023) backbone to learn the parameters of the dynamics. It thus omits the nonlinear autoregressive linearization process. Combined with our custom PyTorch routines for time-parallel filtering and smoothing (Särkkä and García-Fernández, 2020), this approach allows efficient training with the VRKNs principled, uncertainty-capturing objective.

6.4. KalMamba

On a high level, KalMamba embeds a Linear Gaussian State Space Model into a latent space and learns its dynamics model's parameters using a backbone consisting of several mamba layers. It employs a time-parallel Kalman smoother in this latent space to infer latent beliefs for training and acting. By exploiting the associativity of the underlying operations, we can utilize parallel scans for this parallelization. KalMamba employs a tight variational lower bound objective that allows appropriate modeling of uncertainties in noisy, partial-observable systems. We then use a Soft Actor Critic (Haarnoja et al., 2018) approach to learn to act, avoiding autoregressive rollouts for policy learning.

6.4.1. The KalMamba Model

To connect the original, high-dimensional observations $\mathbf{o}_t$ to the latent space for inference, we introduce an intermediate auxiliary observation $\mathbf{w}_t$, which is connected to the latent state by an observation model $q(\mathbf{w}_t|\mathbf{z}_t) = \mathcal{N}\left(\mathbf{w}_t|\mathbf{z}_t, \Sigma_t^{\mathrm{w}}\right)$ (Becker et al., 2019; Shaj et al., 2020). Here, we assume $\mathbf{w}_t$ to be observable and extract it, together with the diagonal observation covariance Σ_t^{w} from the observation using an encoder network; $\left(\mathbf{w}_t, \Sigma_t^{\mathrm{w}}\right) = \phi(\mathbf{o}_t)$. This approach allows us to model the complex dependency between $\mathbf{z}_t$ and $\mathbf{o}_t$ using the encoder while having a simple observation model for inference in the latent space. Opposed to modeling $\mathbf{w}_t$ as a random variable (Fraccaro



$\mathbf{a}_{t-2}$
$\mathbf{w}_{t-1} = \phi(\mathbf{o}_{t-1})$
$\mathbf{a}_{t-1}$
$\mathbf{w}_t = \phi(\mathbf{o}_t)$
$\mathbf{a}_t$
$\mathbf{w}_{t+1} = \phi(\mathbf{o}_{t+1})$
$\mathbf{a}_{t+1}$
Mamba
$\mathbf{m}_{t-1}$
$\mathbf{m}_t$
$\mathbf{m}_{t+1}$
$\mathbf{A}_{t-1}, \mathbf{b}_{t-1}, \mathbf{\Sigma}_{t-1}$
$\mathbf{A}_t, \mathbf{b}_t, \mathbf{\Sigma}_t$
$\mathbf{A}_{t+1}, \mathbf{b}_{t+1}, \mathbf{\Sigma}_{t+1}$

Figure 6.2.: .
Schematic of the Mamba (Gu and Dao, 2023) based backbone to learn the system dynamics. It shares the inference model's encoder $\phi(\mathbf{o}_t)$ and intermediate representation $\mathbf{w}_t$. Each $\mathbf{w}_t$ is then concatenated to the previous action $\mathbf{a}_{t-1}$, fed through a small Neural Network (NN) and given to Mamba model which accumulates information over time and emits a representation $\mathbf{m}_t(\mathbf{o}_{t\leq}, \mathbf{a}_{\leq t-1})$ containing the same information as the filtered belief $q(\mathbf{z}_t|\mathbf{o}_{t\leq}, \mathbf{a}_{\leq t-1})$. We then concatenate each $\mathbf{m}_t$ with the current action $\mathbf{a}_t$ and use another small NN to compute the dynamics parameters $\mathbf{A}_t$, $\mathbf{b}_t$ and Σ_t. This scheme allows us to use the intermediate representation $\mathbf{m}_t$ for regularization and we regularize it towards the filtered belief's mean using a Mahalanobis regularizer (Equation 6.2). Finally, the small NNs include Monte-Carlo Dropout (Gal and Ghahramani, 2016) to model epistemic uncertainty.

et al., 2017; Klushyn et al., 2021), modeling it is observable results in fewer latent variables which simplifies inference and allows direct propagation of the observation uncertainties from the encoder to the state.

We parameterize the dynamics model as

$$p(\mathbf{z}_{t+1}|\mathbf{z}_t, \mathbf{a}_t) = \mathcal{N}\left(\mathbf{z}_{t+1}|\mathbf{A}_t(\mathbf{o}_{\leq t}, \mathbf{a}_{\leq t})\mathbf{z}_t + \mathbf{b}_t(\mathbf{o}_{\leq t}, \mathbf{a}_{\leq t}), \Sigma_t^{\text{dyn}}(\mathbf{o}_{\leq t}, \mathbf{a}_{\leq t})\right) \tag{6.1}$$

where both $\mathbf{A}_t$ and Σ_t^{dyn} are diagonal matrices and we constrain the (eigen)values of $\mathbf{A}$ to be between 0.4 and 0.99. This constraint ensures the resulting dynamics are plausible and stable. This approach effectively linearizes the dynamics parameters $\mathbf{A}_t, \mathbf{b}_t$ and Σ_t^{dyn} around all past observations and actions. Crucially, the resulting dynamics are linear in $\mathbf{z}_t$ enabling the closed-form inference of beliefs using standard Kalman filtering and smoothing.

Parameterizing the dynamics model of Equation 6.1 naively can lead to poor representations, as information can bypass the actual SSM through the linearization backbone. To counter this, we design the backbone architecture as depicted in Figure 6.2. For each timestep, we concatenate $\mathbf{w}_t$ and $\mathbf{a}_{t-1}$, transform each resulting vector using a small neural network, feed it through a Mamba (Gu and Dao, 2023) model and linearly project the output to a vector $\mathbf{m}_t$ of the same dimension as the latent state $\mathbf{z}_t$. Each $\mathbf{m}_t$ now accumulates the same observations and actions used to form the corresponding filtered belief $q(\mathbf{z}_t|\mathbf{o}_{\leq t}, \mathbf{a}_{\leq t-1})$. We then take $\mathbf{m}_t$ and the action

$\mathbf{a}_t$ to compute the dynamics parameters using another small neural network. This bottleneck introduced by $\mathbf{m}_t$ allows us to regularize the model as discussed below. Following (Becker and Neumann, 2022) we further include Monte-Carlo Dropout (Gal and Ghahramani, 2016) into this architecture, as explicitly modeling the epistemic uncertainty is crucial when working with a smoothing inference.

The generative observation model is given by a decoder network $p(\mathbf{o}_t|\mathbf{z}_t)$. The observations are modeled as Gaussian with learned mean and fixed standard deviation. Finally, we assume an initial state distribution $p(\mathbf{z}_0)$ that is a zero mean Gaussian with a learned variance Σ_0.

Given the latent observation model $q(\mathbf{w}_t|\mathbf{z}_t)$, and the shared, pre-computable, linear dynamics model, we can efficiently infer belief states using extended Kalman filtering and smoothing. Recent work (Särkkä and García-Fernández, 2020) shows how to formulate such filtering and smoothing as associative operations amenable to temporal parallelization using associative scans. We implement these operations in PyTorch (Paszke et al., 2019). Similar to S5 (Smith et al., 2022) or Mamba (Gu and Dao, 2023) this implementation yields a logarithmic time complexity, given sufficiently many parallel cores. Additionally, as the dynamics matrix $\mathbf{A}_t$ and all model covariances, i.e., Σ_t^{dym}, Σ_t^{obs}, and Σ_0, are diagonal, the same holds for the covariances of the filtered and smoothed beliefs. Thus we can replace costly matrix operations during Kalman filtering and smoothing with point-wise operations, which further ensures KalMamba's efficiency.

6.4.2. Training the Model

After inserting the state space assumptions of our generative and inference models, the standard variational lower bound to the data marginal log-likelihood (Kingma and Welling, 2013) for a single sequence simplifies to (Becker and Neumann, 2022)

$$\mathcal{L}_{\text{ssm}}(\mathbf{o}_{\leq T}, \mathbf{a}_{\leq T}) = \\ \sum_{t=1}^{T} \left(\mathbb{E}_{q(\mathbf{z}_t|\mathbf{o}_{\leq T},\mathbf{a}_{\leq T})} \left[\log p(\mathbf{o}_t|\mathbf{z}_t)\right] - \mathbb{E}_{q(\mathbf{z}_{t-1}|\mathbf{o}_{\leq T},\mathbf{a}_{\leq T})} \left[\mathrm{KL}\left(q(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{a}_{\geq t-1}, \mathbf{o}_{\geq t}) || p(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{a}_{t-1})\right)\right]\right).$$

Due to the smoothing inference, this lower bound is tight and allows accurate modeling of the underlying system's uncertainties. To evaluate the lower bound we need the smoothed dynamics $q(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{a}_{\geq t-1}, \mathbf{o}_{\geq t})$ whose parameters we can compute given the equations provided in (Becker and Neumann, 2022).

To regularize the Mamba-based backbone used to learn the dynamics, we incentivize $\mathbf{m}_t$ to correspond to the filtered mean using a Mahalonobis distance

$$R(\mathbf{o}_{\leq T}, \mathbf{a}_{\leq T}) = \sum_{t=1}^{T} \left(\mathbf{m}_t(\mathbf{o}_{\leq t}, \mathbf{a}_{\leq t-1} - \boldsymbol{\mu}_t^+\right)^T \left(\Sigma_t^+\right)^{-1} \left(\mathbf{m}_t(\mathbf{o}_{\leq t}, \mathbf{a}_{\leq t-1}) - \boldsymbol{\mu}_t^+\right), \tag{6.2}$$

$\boldsymbol{\mu}_t^+$ and Σ_t^+ denote the mean and variance of the filtered belief $q(\mathbf{z}_t|\mathbf{o}_{\leq t}, \mathbf{a}_{\leq t-1})$. This regularization discourages the model from bypassing information over the Mamba backbone. This mirrors many established models such as the classical extend Kalman Filter (Jazwinski, 1970), which linearize directly around this mean, but still allows associative parallel scanning.

Finally, we add a reward model $p(r_t|\mathbf{z}_t)$, predicting the current reward from the latent state using a small neural network. While this is not strictly necessary for standard policy learning on top of the

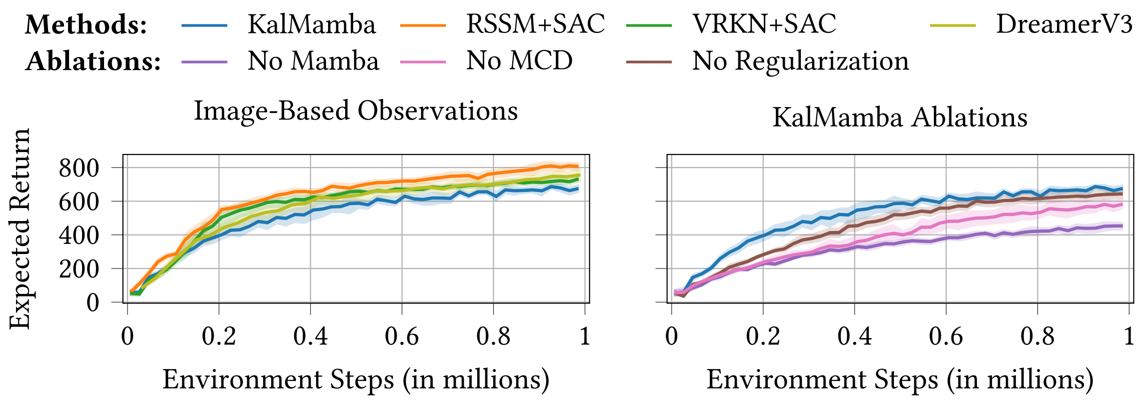


Figure 6.3.: Aggregated expected returns for image-based observations. (**Left**) KalMamba is slightly worse but overall competitive with the different baselines. Combining either baseline SSM with SAC matches or exceeds the performance of DreamerV3. (**Right**) Using Mamba to learn the dynamics is crucial for good model performance. Similarly, both Monte-Carlo Dropout and the regularization loss of Equation 6.2 stabilize the training process and lead to higher expected returns.

representation, it nevertheless helps the model to focus on task-relevant details and learn a good representation for control (Srivastava et al., 2021; Tomar et al., 2023). Including this reward term and the Mahalonobis regularize, the full maximization objective for a single sequence is given as

$$\mathcal{L}_{\text{KalMamba}}(\mathbf{o}_{\leq T}, \mathbf{a}_{\leq T}) = \mathcal{L}_{\text{ssm}}(\mathbf{o}_{\leq T}, \mathbf{a}_{\leq T}) + \mathbb{E}_{q(\mathbf{z}_t|\mathbf{o}_{\leq T}, \mathbf{a}_{\leq T})}\left[\log p(r_t|\mathbf{z}_t)\right] - \alpha R(\mathbf{o}_{\leq T}, \mathbf{a}_{\leq T}).$$

6.4.3. Using KalMamba for Reinforcement Learning

We learn a policy on top of the KalMamba state space representation using Soft Actor Critic (SAC) (Haarnoja et al., 2018). Here, we use the mean of the variational filtered belief $q(\mathbf{z}_t|\mathbf{o}_{\leq t}, \mathbf{a}_{\leq t-1})$ as input for the actor and, together with the action $\mathbf{a}_t$ for the critic. Importantly, we cannot smooth during acting as future observations and actions are unavailable. However, while not directly involved in the loss, the filter belief is still meaningful as the smoothing pass introduces no additional parameters. This inductive bias induces a tight coupling between filtered and smoothed belief that ensures the reasonableness of the former. We independently train the KalMamba world model and SAC by stopping the actor's and critic's gradients from propagating through the world model. We use SAC instead of the typical latent imagination strategy used with RSSMs, the VRKN, and R2I. For all 4 models, rolling out policies in the latent space is autoregressive but these rollouts can be avoided by using a Q-function directly on the inferred belief states.

6.5. Experiments

We evaluate KalMamba on four DeepMind Control (DMC) Suite tasks, namely `cartpole_swingup`, `quardruped_walk`, `walker_walk`, and `walker_run`. We train each task for 1 million environment steps with sequences of length 32 and run 20 evaluation runs every 20, 000 steps. We report the expected return using the mean and 95% stratified bootstrapped confidence intervals (Agarwal

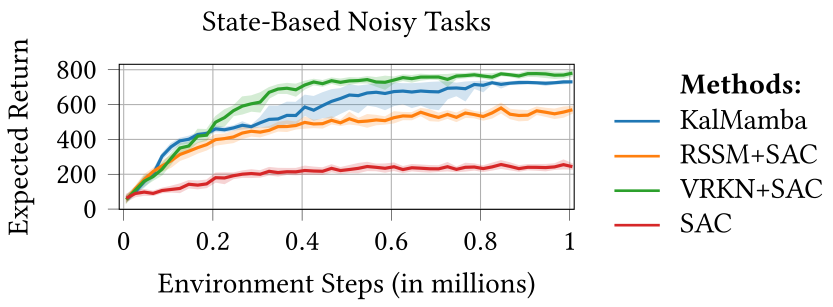


Figure 6.4.: Aggregated expected returns for the state-based noisy tasks. KalMamba clearly outperforms the RSSM while almost matching the VRKN's performance. Naively using SAC is insufficient, which testifies to the increased difficulty due to the noise.

et al., 2021) for 4 seeds per environment. Appendix D.1 provides all hyperparameters. Appendix D.2 provides per-task results for all experiments.

We compare against Recurrent State Space Models (RSSM) and the Variational Recurrent Kalman Network (VRKN). To isolate the effect of the SSMs' representations, we combine both with SAC (Haarnoja et al., 2018) as the RL algorithm, instead of using latent imagination (Hafner et al., 2020).

6.5.1. Standard Image Based Tasks

We first compare on standard image-based observations of the different tasks and include DreamerV3 (Hafner et al., 2023) results for reference. The left side of Figure 6.3 shows the aggregated expected returns. The results indicate that KalMamba is slightly worse, but overall competitive to the two baseline SSMs and DreamerV3, while being parallelizable and thus much more efficient to train. Interestingly, both SSMs work well when combined with SAC, matching or outperforming DreamerV3.

We also conduct ablations for some of the main design choices of KalMamba on the right side of Figure 6.3. No Mamba removes the Mamba layers from the dynamics backbone in Figure 6.2. Similar to the selection mechanism of Mamba (Gu and Dao, 2023) itself, the resulting approach linearizes the dynamics around the current action and observation, instead of all previous observations and actions. The results show that this is insufficient for KalMamba, presumably because it uses only a single SSM layer instead of the stacked layers used by Mamba. Furthermore, No Regularization loss removes the Mahalanobis regularization from the model and No Monte Carlo Dropout removes Monte-Carlo Dropout from the dynamics backbone. Here, the results indicate that regularizing $\mathbf{m}_t$ and explicitly modeling the epistemic uncertainty are crucial for KalMamba's performance.

6.5.2. Low Dimensional Tasks with Observation and Dynamics Noise

To test the models' capabilities under uncertainties, we use the state-based versions of the tasks and add both observation and dynamics noise. The observation noise is sampled from $\mathcal{N}(0, 0.3)$ and added to the observation. The dynamics noise is also sampled from $\mathcal{N}(0, 0.3)$ and added to

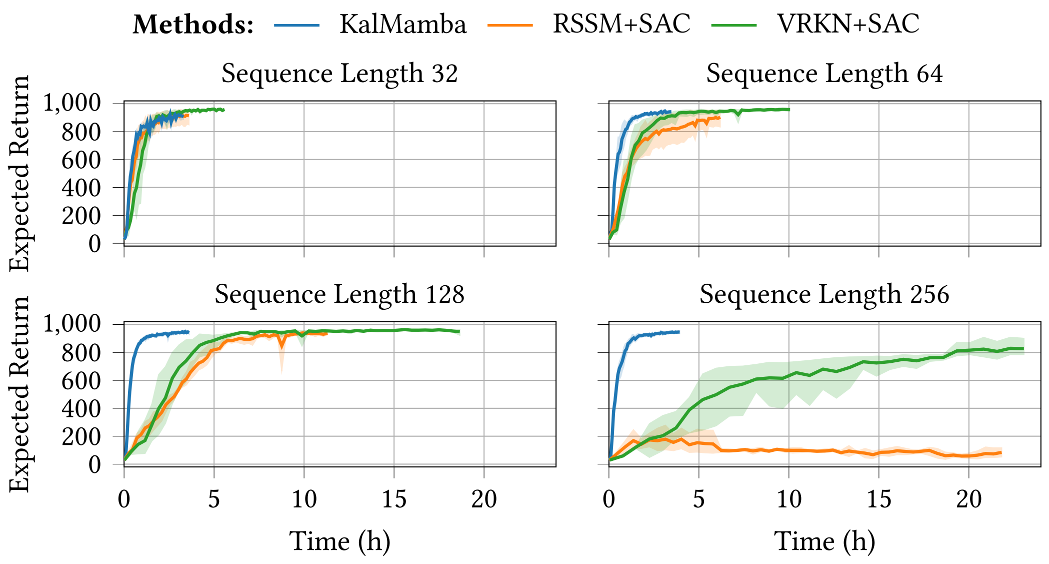


Figure 6.5.: Wall-clock time evaluations on the state-based noisy `walker-walk` for KalMamba, the RSSM, and the VRKN for different training context lengths for 1 million environment steps or up to 24 hours. This time limitation only affected the VRKN training for 256 steps, which reached 650 thousand steps after 24 hours. While all methods work well for short sequences of length 32 (**Top Left**), the efficient parallelization of KalMamba allows it to scale gracefully to and even improve performance for longer sequences of up to 256 steps, where the other methods fail (**Bottom Right**).

the action before execution. However, unlike exploration noise, this addition happens inside the environment and is invisible to the world model and the policy. We include SAC without a world model in our experiments as a baseline to evaluate the difficulty of the resulting tasks.

The results in Figure 6.4 show that naively using SAC fails in the presence of noise. While the RSSM manages to improve performance it is still significantly outperformed by VRKN and KalMamba, which both use the robust smoothing inference scheme. KalMamba needs slightly more environment steps to converge but ultimately almost matches the VRKN's performance while being significantly faster to run.

6.5.3. Runtime Analysis

To show the benefit of KalMamba's efficient parallelization using associative scans, we compare its wall-clock runtime to that of the SSM baselines on the state-based noisy version of `walker-walk` for training sequences of increasing length. The models share a PyTorch implementation and differ only in the SSM. We run each experiment on a single Nvidia Tesla H100 GPU, for up to 1 million steps or 24 hours. Figure 6.5 shows the resulting expected returns. While all models work well for the short sequences of length 32 that are used for the main results above, the training time of the baseline SSMs scales linearly with the sequence length, causing slower convergence and a time-out after 24 hours and 650 thousand environment steps for the VRKN for a length of 256. In comparison, KalMamba shows negligible additional training cost for increased sequence lengths. Further, while the absolute performance of both baselines decreases as the training sequences get longer, KalMamba slightly improves performance when trained on more than 32 steps. These results indicate that KalMamba efficiently utilizes long-term context information through its Mamba

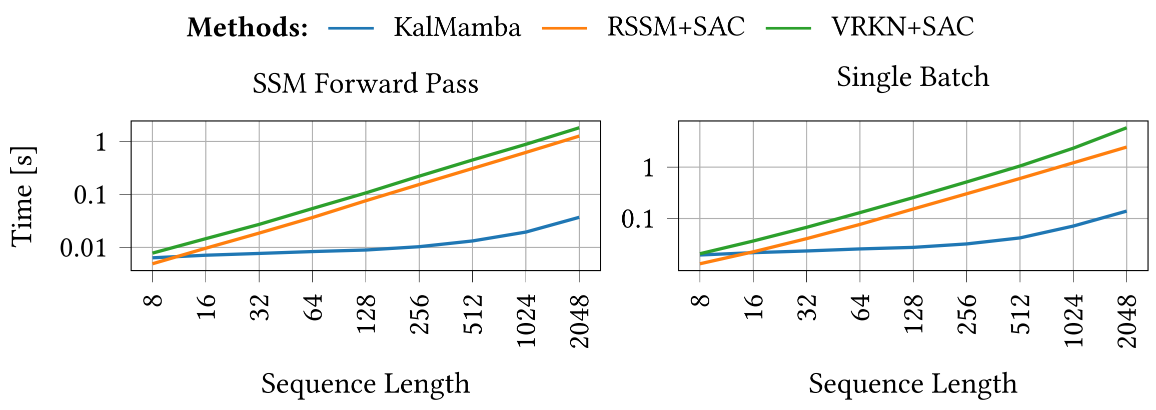


Figure 6.6.: Runtime comparison of KalMamba, the RSSM and the VRKN for (**Left**) a SSM forward pass and (**Right**) a single training batch. While the computational cost of both baseline models scales linearly in the sequence length, KalMamba utilizes associative scans for efficient parallelism and thus near-logarithmic runtime on modern accelerator hardware.

backbone, whereas the dynamics models of the baseline SSMs have difficulty with too-long training sequences.

Investigating this further, we visualize the wall-clock time of a single SSM forward pass and a single training batch for different sequence lengths in Figure 6.6. While both the RSSM and VRKN scale linearly with the sequence length, shows near-logarithmic scaling even for longer sequences thanks to its efficient parallelism. We expect further significant speedups for KalMamba with a potential custom CUDA implementation, similar to Mamba.

6.6. Conclusion

We proposed KalMamba, an efficient State Space Model (SSM) for Reinforcement Learning (RL) under uncertainty. It combines the uncertainty awareness of probabilistic SSMs with recent deterministic SSMs' scalability by embedding a :inear Gaussian SSM into a latent space. We use Mamba (Gu and Dao, 2023) to learn the linearized dynamics in this latent space efficiently. Inference in this SSM amounts to standard Kalman filtering and smoothing and is amenable to full parallelization using associative scans (Särkkä and García-Fernández, 2020). During model learning, this allows time-parallel estimation of smoothed belief states, which allows the efficient usage of principled objectives for uncertainty estimation, especially over long sequences.

Our experiments on low-dimensional states and image observations indicate that KalMamba can match the performance of state-of-the-art stochastic SSMs for RL under uncertainty. In terms of both runtime and performance, KalMamba scales more gracefully to longer training sequences. In particular, its performance improves with sequence length while it degrades for the baseline SSMs.

Limitations and Future Work. The present work explores KalMamba's potential in small-scale experiments, but a more elaborate evaluation of diverse, more realistic tasks would help to explore our method's strengths and weaknesses. A thorough comparison of recent baselines is needed to contextualize KalMamba against existing time-efficient SSMs with simplified, non-smoothing inference schemes (Samsami et al., 2024). We additionally aim to refine KalMamba to improve its performance across wide-ranging tasks. In this context, we plan to model the state with a complex-valued random variable to expand the range of dynamics models that can be learned. Other ideas include improving the regularization of the Mamba backbone and investigating more advanced policy learning methods that make use of the uncertainty in the filtered beliefs.

Contribution Statement. *I devised and derived the KalMamba model, implemented it including the efficient parallel scanning for filtering and smoothing, devised and ran the large majority of the evaluations, and wrote the large majority of the article. NF helped suggested experiments and evaluations, and helped with writing the article. GN supervised the project.*

7. Combining Reconstruction and Contrastive Methods for Multi-Sensor Representations in RL

We will now turn to the third research question of this thesis, which concerns how to combine information from multiple sensors with highly different properties into a single, concise representation as a basis for Reinforcement Learning (RL). In such settings, an agent needs a world model that has learned to fuse information from all available sensors, integrate relevant information from the past, and discard irrelevant details. This chapter will focus on answering the third research question of this thesis:

Q3 How can we use probabilistic world models to combine information from multiple sensors with highly different properties into a single, concise representation as a basis for RL?

Our solution, Contrastive Reconstructive Aggregated representation Learning (CoRAL), is a unified framework for state space representation learning that enables combining standard variational autoencoding with contrastive approaches, which maximize mutual information and allows tailoring a loss function to the properties of the given sensors.

Originally, this research was motivated by the observation that the vast majority of RL approaches either directly work on fully observable states or only consider images as observations. However, many real-world embodied agents, such as robots or autonomous vehicles, use a variety of sensors, including cameras, joint encoders, force-torque sensors, IMUs, and more. Of these sensors, some emit concise, noise-free, and informative signals, such as joint encoders. Others are noisy and only provide partial information about the environment state, such as cameras. In terms of probabilistic modeling, State Space Models (SSMs) are a natural choice for such multi-sensor settings, and they are heavily used in more traditional approaches (Khaleghi et al., 2013) to fuse information into a single state estimate.

The following was published as Combining Reconstruction and Contrastive Methods for Multimodal Representations in RL *in the Reinforcement Learning Journal and presented at the accompanying first Reinforcement Learning Conference (RLC 2024) (Philipp Becker, Sebastian Mossburger, Fabian Otto, and Gerhard Neumann, 2024b). For this chapter, the contents of the original publication were restructured, and the wording and formulation were slightly revised in several places.*

The original publication used the term multimodal *to refer to the combination of multiple input modalities, which reflects the common meaning of the term* multimodal *in a deep learning context at the time of writing both the paper and thesis. However, I replaced it with the term* multi-sensor *for this thesis to avoid confusion with the concepts of multimodal distributions and multimodal behavior.*

7.1. Introduction

Most representation learning approaches for Reinforcement Learning (RL) (Hafner et al., 2020, 2021, 2023; Laskin et al., 2020; Lee et al., 2020; Yarats et al., 2021b; Zhang et al., 2020; Zhu et al., 2023; Deng et al., 2022) focus on images. Here, the challenge lies in compressing relevant information while not getting distracted by potentially irrelevant aspects. Yet, most agents in realistic scenarios can directly observe their internal states using sensors in the actuators, inertial measurement units, and force and torque sensors. Including this low-dimensional and concise proprioceptive sensing in representation learning can improve representation quality and downstream RL performance. For such multi-sensor representations, State Space Models (Murphy, 2012) are a natural choice as they lend themselves to accumulating information across multiple sensors and time. Previous works suggest using either reconstruction (Hafner et al., 2019, 2021) or contrastive methods (Hafner et al., 2020; Ma et al., 2020; Nguyen et al., 2021; Srivastava et al., 2021), both with their individual strengths and weaknesses. While reconstruction provides an informative learning signal, it may fail to learn good representations if observations are noisy or contain distracting elements (Zhang et al., 2020; Ma et al., 2020; Deng et al., 2022). In such cases, contrastive methods can ignore irrelevant parts of the observation and still learn valuable representations. However, they are prone to representation collapse and often struggle to learn accurate dynamics (Ma et al., 2020). We argue that the different properties of sensors, such as images and proprioception, suggest using different self-supervised loss functions for each modality.

We propose Contrastive Reconstructive Aggregated representation Learning (CoRAL) to combine contrastive and reconstruction-based approaches. CoRAL builds on state space representations and allows us to select the best-suited loss function for each modality, for example, reconstruction-based loss functions for concise, low-dimensional proprioception and contrastive losses for images with distractions. Learning such state space representations can be theoretically motivated using a Variational Inference (Hafner et al., 2020; Ma et al., 2020) or a predictive coding (Oord et al., 2018; Nguyen et al., 2021; Srivastava et al., 2021) viewpoint, which results in two instances of CoRAL. For both paradigms, CoRAL relies on the insight that we can replace likelihood-based reconstruction terms with contrastive losses based on mutual information, which allows for a principled combination of the two (Hafner et al., 2020; Ma et al., 2020). Figure 7.1 provides an overview of the approach.

We integrate CoRAL into model-free and model-based RL to systematically assess the effects of learning multi-sensor representations by selecting appropriate losses. We evaluate on DeepMind Control (DMC) Suite (Tassa et al., 2018) tasks which we make more difficult by adding Video Backgrounds (Zhang et al., 2020; Nguyen et al., 2021) and Occlusions. Furthermore, we use a new Locomotion suite where agents must fuse proprioception and egocentric vision to move while navigating obstacles. Finally, we consider a novel challenging Manipulation suite consisting of static and mobile manipulation tasks with varying object geometries, built on ManiSkill2 (Gu et al., 2023). Here, the agents must combine proprioception and different visual modalities, such as color and depth, to move, navigate, and interact with varying objects in visually realistic environments. These experiments show that learning multi-sensor representations using the best-suited loss for each modality improves over other methods combining both modalities, such as representation learning with a single loss and concatenating image representations with proprioception. CoRAL tends to work better than recent baselines on the Video Background and Occlusion tasks and allows significant performance gains in the challenging Locomotion and Manipulation tasks. Furthermore, CoRAL significantly improves model-based approaches with contrastive image representations,

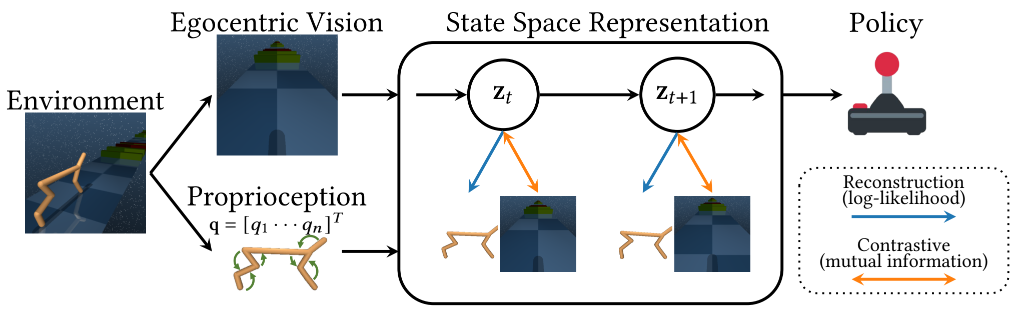

Environment
Egocentric Vision
Proprioception
$q = [q_1 \cdots q_n]^T$
State Space Representation
z_t
z_{t+1}
Policy
Reconstruction (log-likelihood)
Contrastive (mutual information)

Figure 7.1.: Contrastive Reconstructive Aggregated representation Learning (CoRAL) learns multi-sensor state space representations of all available sensors using a combination of reconstruction-based and contrastive objectives. Building on the insight that we can exchange likelihood-based reconstruction with contrastive approaches using mutual information, allows us to choose an appropriate loss function for each modality. Motivated by both a variational and predictive coding viewpoint, CoRAL helps model-free and model-based agents to excel in challenging tasks that require information fusion from sensors with different properties such as images and proprioception.

which are known to perform worse than reconstruction-based approaches (Hafner et al., 2020; Ma et al., 2020). Finally, we show the strengths of both instances of CoRAL. Variational CoRAL excels in tasks where the main challenge is filtering out irrelevant distractions from images, while Predictive CoRAL performs better in tasks that require propagating information over many timesteps.

To summarize our contributions: **(i)** We propose CoRAL, a general framework for multi-sensor representation learning for RL which allows using the best-suited self-supervised loss for each modality using the interchangeability of likelihood-based reconstruction and contrastive losses based on mutual information. **(ii)**We instantiate two versions of CoRAL using state space representations, namely Variational-CoRAL and Predictive-CoRAL, which are inspired by variational and contrastive predictive coding viewpoints, respectively. **(iii)** We systematically show their effectiveness on a diverse set of 26 tasks, across the Video Backgrounds, Occlusions, Locomotion, and Manipulation suites.

7.2. Related Work

Representations for Reinforcement Learning. Many recent approaches use ideas from generative (Wahlström et al., 2015; Watter et al., 2015; Banijamali et al., 2018; Lee et al., 2020; Yarats et al., 2021b) and self-supervised representation learning (Zhang et al., 2020; Laskin et al., 2020; Yarats et al., 2021a; Stooke et al., 2021; You et al., 2022) to improve performance, sample efficiency, and generalization of RL from images. Those based on Recurrent State Space Models (RSSMs) (Hafner et al., 2019) are particularly relevant for this work. When proposing the RSSM, Hafner et al. (2019) used a generative approach. They formulated their objective as auto-encoding Variational Inference (Kingma and Welling, 2013), which trains the representation by reconstructing observations. Such reconstruction-based approaches have limitations with observations containing noise or many task-irrelevant details. As a remedy, Hafner et al. (2020) proposed a contrastive alternative based on mutual information and the InfoNCE estimator (Poole et al., 2019). Ma et al. (2020) refined this approach and improved results by modifying the policy learning mechanism. Using a different motivation, namely contrastive predictive coding (Oord et al., 2018), Okada and Taniguchi (2021);

Nguyen et al. (2021); Srivastava et al. (2021); Okada and Taniguchi (2022) proposed alternative contrastive learning objectives for RSSMs. In this work, we leverage the variational and predictive coding paradigms and show that CoRAL improves performance for both. Fu et al. (2021); Wang et al. (2022) propose further factorizing the RSSM's latent variable to disentangle task-relevant and task-irrelevant information. However, unlike contrastive approaches, they explicitly model the task-irrelevant parts instead of ignoring them, which can impede performance if the distracting elements become too complex to model. Zhu et al. (2023) propose a relaxed variational information bottleneck (Alemi et al., 2016) approach which trains RSSMs solely by predicting rewards and enforcing posterior predictability using a KL term. Other recent approaches for learning RSSMs include using prototypical representations (Deng et al., 2022) or masked reconstruction (Seo et al., 2022).

Sensor Fusion in Reinforcement Learning. Many application-driven approaches to visual RL for robots use proprioception to solve their specific tasks (Finn et al., 2016; Levine et al., 2016; Kalashnikov et al., 2018; Xiao et al., 2022; Fu et al., 2022). Yet, they usually do not use explicit representation learning or concatenate image representations and proprioception. Several notable exceptions use RSSMs with images and proprioception (Wu et al., 2022; Becker and Neumann, 2022; Hafner et al., 2022, 2023). Furthermore, Seo et al. (2023) learn world models using multiple images from different viewpoints. However, all these approaches focus on purely reconstruction-based representation learning. Srivastava et al. (2021) use images and proprioceptive information using contrastive predictive coding for both modalities. Opposed to all of these works, we propose combining contrastive approaches with reconstruction.

Multi-Sensor Representation Learning. Representation learning from multiple modalities has widespread applications in general machine learning, where methods such as CLIP (Radford et al., 2021) combine language concepts with the semantic knowledge of images and allow language-based image generation (Ramesh et al., 2022). For robotics, Brohan et al. (2022); Mees et al. (2022); Driess et al. (2023); Shridhar et al. (2022, 2023) combine language models with the robot's perception for natural language-guided manipulation tasks using Imitation Learning. In contrast, CoRAL assumes an online RL setting and focuses on different modalities, namely images and proprioception.

7.3. Combining Contrastive Approaches and Reconstruction for State Space Representations

Given trajectories of observations $\mathbf{o}_{\leq T} = (\mathbf{o}_t)_{t=1\cdots T}$ and actions $\mathbf{a}_{\leq T} = (\mathbf{a}_t)_{t=1\cdots T}$ we aim to learn a state representation that is well suited for RL. We assume the observations stem from K different sensors, $\mathbf{o}_t = \left(\mathbf{o}_t^{(k)}\right)_{k=1\cdots K}$, where the individual $\mathbf{o}_t^{(k)}$ only contain partial information about the state. Further, even $\mathbf{o}_t$ may not contain all necessary information for optimal acting, i. e., the environment is partially observable, and the representation has to accumulate information over time. Our goal is to learn a concise, low dimensional representation $\phi(\mathbf{o}_{\leq t}, \mathbf{a}_{\leq t-1})$ that accumulates all relevant information until time step t. We provide this representation to a policy $\pi(\mathbf{a}_t | \phi(\mathbf{o}_{\leq t}, \mathbf{a}_{\leq t-1}))$ which aims to maximize the expected return in a given RL problem. In this setting, the policy's final return and the sample complexity of the entire system determine what constitutes a good representation.

State Space Models (SSMs) (Murphy, 2012) naturally lend themselves to sensor fusion and information accumulation problems. We assume a latent state variable, $\mathbf{z}_t$, which evolves according to a Markovian dynamics $p(\mathbf{z}_{t+1}|\mathbf{z}_t, \mathbf{a}_t)$ given an action $\mathbf{a}_t$. Furthermore, we assume the K observations at each time step are conditionally independent given the latent state, resulting in an observation model $p(\mathbf{o}_t|\mathbf{z}_t) = \prod_{k=1}^{K} p^{(k)}(\mathbf{o}_t^{(k)}|\mathbf{z}_t)$. The initial state is distributed according to $p(\mathbf{z}_0)$. Here, the belief over the latent state, taking into account all previous actions as well as previous and current observations $p(\mathbf{z}_t|\mathbf{a}_{\leq t-1}, \mathbf{o}_{\leq t})$ can be used as the representation. Yet, computing $p(\mathbf{z}_t|\mathbf{a}_{\leq t-1}, \mathbf{o}_{\leq t})$ analytically is intractable for models of relevant complexity and we use a variational approximation $\phi(\mathbf{o}_{\leq t}, \mathbf{a}_{\leq t-1}) \hat{=} q(\mathbf{z}_t|\mathbf{a}_{\leq t-1}, \mathbf{o}_{\leq t})$. This variational approximation also plays an integral part during training and is thus readily available as input for the policy.

We instantiate the generative SSM and the variational distribution using a Recurrent State Space Model (RSSM) (Hafner et al., 2019), which splits the latent state $\mathbf{z}_t$ into a stochastic and a deterministic part. Following Hafner et al. (2019, 2020), we assume the stochastic part of the RSSM's latent state to be Gaussian. While the original RSSM only has a single observation model $p(\mathbf{o}_t|\mathbf{z}_t)$, we extend it to K models, one for each observation modality. The variational distribution takes the deterministic part of the state together with the K observations $\mathbf{o}_t = \{\mathbf{o}_t^{(k)}\}_{k=1\cdots K}$ and factorizes as $q(\mathbf{z}_{\leq t}|\mathbf{o}_{\leq t}, \mathbf{a}_{\leq t-1}) = \prod_{t=1}^{T} q(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{a}_{t-1}, \mathbf{o}_t)$. To account for multiple observations instead of one, we first encode each observation individually using a set of K encoders, concatenate their outputs, and provide the result to the RSSM. Finally, we also learn a reward model $p(r_t|\mathbf{z}_t)$ to predict the reward from the representation. Following the findings of Srivastava et al. (2021) and Tomar et al. (2023) we also include reward prediction to learn the representations for model-free agents.

7.3.1. Learning the State Space Representation

We propose to combine reconstruction-based and contrastive approaches to train our representations. Training RSSMs can be based on either a variational viewpoint (Hafner et al., 2020; Ma et al., 2020) or a contrastive predictive coding (Oord et al., 2018) viewpoint (Nguyen et al., 2021; Srivastava et al., 2021). We investigate both approaches, as neither decisively outperforms the other.

Originally, Hafner et al. (2019) proposed leveraging a fully generative approach for RSSMs. Building on the stochastic variational autoencoding Bayes framework (Kingma and Welling, 2013; Sohn et al., 2015), they derive a variational lower-bound objective

$$\begin{aligned}&\mathbb{E}_{p(\mathbf{o}_{\leq T}, \mathbf{a}_{\leq T})}\left[\log p(\mathbf{o}_{\leq T}|\mathbf{a}_{\leq T})\right]\\ &\geq \sum_{t=1}^{T} \mathbb{E}_{\hat{q}(\cdot)}\left[\log p(\mathbf{o}_t|\mathbf{z}_t) - \mathrm{KL}\left(q(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{a}_{t-1}, \mathbf{o}_t) || p(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{a}_{t-1})\right)\right],\end{aligned}$$

where $\hat{q}(\cdot) = q(\mathbf{z}_{t-1,t}|\mathbf{o}_{\leq t}, \mathbf{a}_{\leq t})p(\mathbf{o}_{\leq t}, \mathbf{a}_{\leq t})$, i. e.,the variational distribution and sub-trajectories from a replay buffer. After inserting our assumption that each observation factorizes into K independent observations, i.e., $\log p(\mathbf{o}_t|\mathbf{z}_t) = \sum_{k=1}^{K} \log p^{(k)}(\mathbf{o}_t^{(k)}|\mathbf{z}_t)$, and adding a term for reward prediction, this results in

$$\sum_{t=1}^{T} \mathbb{E}_{\hat{q}(\cdot)}\left[\sum_{k=1}^{K} \log p^{(k)}(\mathbf{o}_t^{(k)}|\mathbf{z}_t) + \log p(r_t|\mathbf{z}_t) - \mathrm{KL}\left(q(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{a}_{t-1}, \mathbf{o}_t) || p(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{a}_{t-1})\right)\right]. \quad (7.1)$$

Optimizing this bound using the reparameterization trick (Kingma and Welling, 2013; Rezende et al., 2014) and stochastic gradient descent simultaneously trains the variational distribution and all parts of the generative model. While this approach can be highly effective, reconstructing high-dimensional, noisy observations can also cause issues. First, it requires introducing large observation models. These observation models are unessential for the downstream task and are usually discarded after training. Second, the reconstruction forces the model to capture all details of the observations, which can lead to highly suboptimal representations if images contain task-irrelevant distractions.

Contrastive Variational Learning (CV) can remedy these problems. To introduce contrastive terms, we replace the individual reconstruction terms in Equation 7.1 with mutual information (MI) terms $I(\mathbf{o}_t^{(k)}, \mathbf{z}_t)$ by adding and subtracting $\log p^{(k)}(\mathbf{o}^{(k)})$ (Hafner et al., 2020; Ma et al., 2020)

$$\mathbb{E}_{\hat{q}(\cdot)}\left[\log p^{(k)}(\mathbf{o}_t^{(k)}|\mathbf{z}_t)\right] = \mathbb{E}_{\hat{q}(\cdot)}\left[\log \frac{p^{(k)}(\mathbf{o}_t^{(k)}|\mathbf{z}_t)}{p(\mathbf{o}_t^{(k)})} + \log p(\mathbf{o}_t^{(k)})\right] = I(\mathbf{o}_t^{(k)}, \mathbf{z}_t) + c. \tag{7.2}$$

Intuitively, the MI measures how informative a given latent state is about the corresponding observations. Thus, maximizing it leads to similar latent states for similar sequences of observations and actions. While we cannot analytically compute the MI, we can estimate it using the InfoNCE bound (Oord et al., 2018; Poole et al., 2019). Doing so eliminates the need for generative reconstruction. It instead only requires a discriminative approach based on a score function $f_v^{(k)}(\mathbf{o}_t^{(k)}, \mathbf{z}_t) \mapsto \mathbb{R}_+$. This score function measures the compatibility of pairs of observations and latent states. It shares large parts of its parameters with the RSSM. We refer to Appendix E.1 for details on the exact parameterization. This methodology allows the mixing of reconstruction and mutual information terms for the individual sensors, resulting in a generalization of Equation 7.1,

$$\sum_{t=1}^{T}\sum_{k=1}^{K} \mathcal{L}_v^{(k)}(\mathbf{o}_t^{(k)}, \mathbf{z}_t) + \mathbb{E}_{\hat{q}(\cdot)}\left[\log p(r_t|\mathbf{z}_t) - \mathrm{KL}\left(q(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{a}_{t-1}, \mathbf{o}_t)||p(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{a}_{t-1})\right)\right]. \tag{7.3}$$

Here $\mathcal{L}_v^{(k)}$ is either $\mathbb{E}_{\hat{q}(\cdot)}\left[\log p(\mathbf{o}_t^{(k)}|\mathbf{z}_t)\right]$ or $I(\mathbf{o}_t^{(k)}, \mathbf{z}_t)$. As we show in Section 7.5 choosing the terms corresponding to the properties of the corresponding modality can often improve performance.

Contrastive Predictive Coding (CPC) (Oord et al., 2018) provides an alternative to the variational approach. The idea is to maximize the MI between the previous latent variable $\mathbf{z}_{t-1}$ and the observation $\mathbf{o}^{(k)}$, i. e., $I(\mathbf{o}_t^{(k)}, \mathbf{z}_{t-1})$. While this approach seems similar to contrastive variational learning, we use the previous latent state $\mathbf{z}_{t-1}$ instead of the current $\mathbf{z}_t$ to estimate the MI. Thus, we explicitly predict one time step ahead to compute the loss. As we use the RSSM's dynamics model for the prediction, this formalism provides a training signal to the dynamics model. However, Levine et al. (2019); Shu et al. (2020); Nguyen et al. (2021) discuss how this signal alone is insufficient for model-based RL. Srivastava et al. (2021) show that similar ideas also benefit model-free RL and we follow their approach by regularizing the objective using KL-term from Equation 7.1 weighted with a small factor β. Additionally, we can turn individual contrastive MI terms into reconstruction terms for suitable sensor modalities by reversing the principle of Equation 7.2. Including reward prediction, this results in the following maximization objective

$$\sum_{t=1}^{T}\sum_{k=1}^{K} \mathcal{L}_p^{(k)}(\mathbf{o}_{t+1}^{(k)}, \mathbf{z}_t) + \mathbb{E}_{\hat{q}(\cdot)}\left[\log p(r_t|\mathbf{z}_t) - \beta\mathrm{KL}\left(q(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{a}_{t-1}, \mathbf{o}_t)||p(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{a}_{t-1})\right)\right], \tag{7.4}$$

where $\mathcal{L}_p^{(k)}$ is either the one-step ahead likelihood $\mathbb{E}_{\hat{q}(\cdot)}\left[\log p(\mathbf{o}_t^{(k)}|\mathbf{z}_{t-1})\right]$ or an InfoNCE estimate of $I(\mathbf{o}_t^{(k)}, \mathbf{z}_{t-1})$ using a score function $f_p^{(k)}(\mathbf{o}_t^{(k)}, \mathbf{z}_{t-1}) \mapsto \mathbb{R}_+$. From an implementation viewpoint, the resulting approach differs only slightly from the variational contrastive one. For CPC approaches, we use a sample from the RSSM's dynamics $p(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{a}_{t-1})$ and for contrastive variational approaches we use a sample from the variational distribution $q(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{a}_{t-1}, \mathbf{o}_t)$ as input to the score function or decoder.

Estimating Mutual Information with InfoNCE. We estimate the mutual information (MI) using b mini-batches of sub-sequences of length l. After computing the latent estimates, we get $N = b \cdot l$ pairs $(\mathbf{o}_i, \mathbf{z}_i)$, i. e., we use both samples from the elements of the batch as well as all the other time steps within the sequence as negative samples. Using those, the symmetry of MI, the InfoNCE bound (Poole et al., 2019), and either $f = f_v^{(k)}$ or $f = f_p^{(k)}$, we can estimate the MI as

$$I(\mathbf{o}_i, \mathbf{z}_i) \geq 0.5 \left(\sum_{i=1}^{N} \log \frac{f(\mathbf{o}_i, \mathbf{z}_i)}{\sum_{j=1}^{N} f(\mathbf{o}_i, \mathbf{z}_j)} + \log \frac{f(\mathbf{o}_i, \mathbf{z}_i)}{\sum_{j=1}^{N} f(\mathbf{o}_j, \mathbf{z}_i)} \right).$$

The score function for the contrastive variational objective $f_v^{(k)}$ is given as

$$f_v^{(k)}(\mathbf{o}_t^{(k)}, \mathbf{z}_t) = \exp\left(\frac{1}{\lambda} \rho_o \left(\psi_{\text{obs}}^{(k)}(\mathbf{o}_t^{(k)}) \right)^T \rho_z(\mathbf{z}_t) \right),$$

where $\psi_{\text{obs}}^{(k)}$ is the RSSM's encoder and λ is a learnable inverse temperature parameter. ρ_o and ρ_z are projections that project the embedded observation and latent state a 50 dimension embedding. ρ_o is only a single linear layer while ρ_z is a 2×256 fully connected NN. Both use LayerNorm (Ba et al., 2016) at the output.

Similarly, the score function for the contrastive predictive objective $f_p^{(k)}$ is given as

$$f_p^{(k)}(\mathbf{o}_t^{(k)}, \mathbf{z}_{t-1}) = \exp\left(\frac{1}{\lambda} \rho_o \left(\psi_{\text{obs}}^{(k)}(\mathbf{o}_t^{(k)}) \right)^T \rho_z(\phi_{\text{dyn}}(\mathbf{z}_{t-1}, \cdot)) \right).$$

Here ϕ_{dyn} is the RSSM's dynamics model, which propagates the latent state forward in time and is now explicitly part of the score function. All other components are the same as in the contrastive variational case.

7.3.2. Learning to Act Based on the Representation

Our representations are amenable to both model-free and model-based Reinforcement Learning. For the former, we use Soft Actor-Critic (SAC) (Haarnoja et al., 2018) on top of the representation by providing the deterministic part of the latent state and the mean of the stochastic part as input to both the actor and the critic. For the latter, we use latent imagination (Hafner et al., 2020), which propagates gradients through the learned dynamics model to optimize the actor. In both cases, we alternatingly update the RSSM, actor, and critic for several steps before collecting a new sequence in the environment. The RSSM uses only the representation learning loss and gets neither gradients from the actor nor the critic.

7.4. Experiment Setup

A reference implementation of CoRAL and all considered tasks is available at:

`https://github.com/pbecker93/CoRAL/`

Building on the previously introduced methodology, we build two versions of Contrastive Reconstructive Aggregated representation Learning (CoRAL) differing in the state space representation objective. Variational CoRAL (V-CoRAL), using the variational objective (Equation 7.3) and Predictive CoRAL (P-CoRAL), using the predictive coding objective (Equation 7.4). We evaluate the performance of CoRAL by using it for downstream online RL and assessing the average expected return or success rate.

7.4.1. Baselines and Ablations

To show the benefits of combining contrastive and reconstruction-based objectives, we compare with ablative variants that use the same loss for both modalities (Same-Loss), the naive approach of concatenating proprioception to image representations (Concat) and using only the image (Img-Only). We consider the contrastive variational (CV) and the contrastive predictive coding (CPC) paradigm for each of these approaches. For reference, we also include reconstruction-based (Recon.) approaches (Equation 7.1). Furthermore, we use SAC (Haarnoja et al., 2018) on only the proprioception (ProprioSAC), to show that proprioception alone is insufficient to solve the tasks. Finally, we consider the model-free DrQ-v2 (Yarats et al., 2022) and model-based RePo (Zhu et al., 2023) as baselines to demonstrate the competitiveness of our approach. We extend both to also use proprioception and refer to the resulting approaches as DrQ-v2(I+P) and RePo(I+P) respectively.

Some of our ablations are similar to related approaches. The model-based Img-Only ablation with reconstruction loss, is very similar to Dreamer-v1(Hafner et al., 2020). It differs from the Dreamer-v1 (Hafner et al., 2020) in using the KL-balancing introduced in (Hafner et al., 2021) and in regularizing the value function towards its own exponential moving average, as introduced in (Hafner et al., 2023). However, there are considerable differences between the contrastive version of Dreamer-v1(Hafner et al., 2020) and the contrastive variational Img-Only ablation. In particular, those regard the exact form of mutual information estimation and the use of image augmentations.

The model-free contrastive predictive Img-Only and Same-Loss ablations are similar to the approach of Srivastava et al. (2021). The main difference is that Srivastava et al. (2021) includes the critic's gradients when updating the representation while in our setting no gradients flow from the actor or the critic to the representation. Furthermore, we did not include the inverse dynamics objective used by Srivastava et al. (2021) as we did not find it to be helpful. Additionally, we adapted some hyperparameters to match those of our other approaches.

Environment	Proprioceptive	Non-Proprioceptive
Ball In Cup	cup position and velocity	ball position and velocity
Cartpole	cart position and velocity	pole angle and velocity
Cheetah	joint positions and velocities	global pose and velocity
Reacher	reacher position and velocity	distance to target
Quadruped	joint positions and velocities	global pose + velocity, forces
Walker	orientations and velocities of links	global pose and velocity, height above ground

Table 7.1.: Splits of the entire system state into proprioceptive and non-proprioceptive parts for the DeepMind Control Suite environments.

7.4.2. Modified Deep Mind Control Suite Tasks

We use 7 tasks from the DeepMind Control Suite (DMC) (Tassa et al., 2020) that cover a wide range of challenges, namely `Ball-in-Cup Catch`, `Cartpole Swingup`, `Cheetah Run`, `Reacher Easy`, `Walker Walk`, `Walker Run`, and `Quadruped Walk`. We split their states into proprioceptive and non-proprioceptive entries, where the proprioception only contains partial information about the state. The remaining information has to be inferred from images. For example, in `Ball-in-Cup Catch` the cup's state is proprioceptive while the ball's state is not. Table 7.1 lists the splits for the remaining tasks. Additionally, we create two suites by adding Video Backgrounds or Occlusions for all seven tasks.

Video Backgrounds. Following (Zhang et al., 2020; Fu et al., 2021; Nguyen et al., 2021; Deng et al., 2022; Wang et al., 2022; Zhu et al., 2023) we render videos from the `driving car` class of the Kinetics400 dataset (Kay et al., 2017) behind the agents to add a natural video background. However, previous works implement this idea in two distinct ways. Nguyen et al. (2021) and Deng et al. (2022) use color images as background and pick a random sub-sequence of a random video for each environment rollout. They adhere to the train-validation split of the Kinetcs400 dataset, using training videos for representation and policy learning and validation videos during evaluation. Zhang et al. (2020); Fu et al. (2021); Wang et al. (2022); Zhu et al. (2023), according to the official implementations, instead work with gray-scale images and sample a single background video from the train set once during initialization of the environment. They do not sample a new video during the environment reset, thus all training sequences have the same background video. We follow the first approach, as we believe it mimics a more realistic scenario of always changing and colored natural background.

Occlusions. Following (Becker and Neumann, 2022), we render slow-moving disks over the original observations to occlude parts of the observation. The speed of the disks makes memory necessary, as they can occlude relevant aspects for multiple consecutive timesteps.

Figure 7.2 shows example images the modifications. For both suites, the challenge is to learn representations that filter out irrelevant visual details while focusing on relevant aspects. Occlusions also tests the approaches' capabilities to maintain a consistent representation across time under partial observability, considerably increasing the task's difficulty.

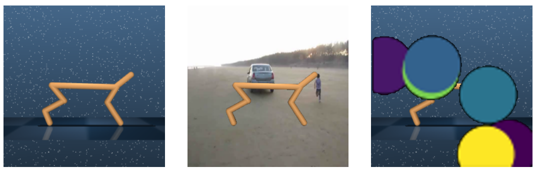


Figure 7.2.: Example images of the `Cheetah Run` task. **Left:** Standard image. **Middle**: Image with natural video background. **Right:** Image with occlusions.

Environment	Proprioceptive	Non-Proprioceptive
Ant	joint position and velocity global velocities	wall positions global position
Hurdle Cheetah	joint positions and velocities global velocity	hurdle positions and heights global position
Hurdle Walker	orientations and velocities of links	hurdle positions and height global position and velocity
Quadruped (Escape)	joint positions and velocities, torso orientation and velocity, imu, forces, and torques at joints	Information about terrain

Table 7.2.: Splits of the entire system state into proprioceptive and non-proprioceptive parts for the Locomotion Suite. Some of the agents (Cheetah, Walker, Quadruped) require more proprioceptive information for the locomotion tasks with an egocentric vision than for the standard tasks with images from an external perspective.

7.4.3. Locomotion Suite

Building on the DeepMind Control Suite (Tassa et al., 2020), we introduce a novel Locomotion suite consisting of six tasks: `Hurdle-Cheetah Run`, `Hurdle-Walker Walk`, `Hurdle-Walker Run`, `Ant-Empty`, `Ant-Walls` and `Quadruped Escape`. Table 7.2 shows the splits into proprioceptive and non-proprioceptive parts. Figure 7.3 displays all environments in the suite. All tasks include obstacles that have to be localized through egocentric vision to be avoided. As the agents cannot observe themselves from the egocentric perspective, they additionally need proprioception. These tasks test the representations' ability to combine information from both sources to enable successful navigation and movement.

Both `Ant` tasks build on the locomotion functionality introduced into the DeepMind Control suite by (Tassa et al., 2020). For `Ant Empty`, we only use an empty corridor, which makes this the easiest task in our locomotion suite. For `Ant Walls`, we randomly generate walls inside the corridor, and the agent has to avoid those to achieve its goal, i.e., running through the corridor as fast as possible.

For the `Hurdle Cheetah` and `Hurdle Walker` tasks we modified the standard `Cheetah Run`, `Walker Walk`, and `Walker Run` tasks by introducing "hurdles" over which the agent has to step to move forward. The hurdles' positions, heights, and colors are reset randomly for each episode, and the agent has to perceive them using egocentric vision. For this vision, we added a camera in the head of the Cheetah and Walker. Note that the hurdle color is not relevant to avoid them and thus introduces irrelevant information that needs to be captured by reconstruction-based approaches.

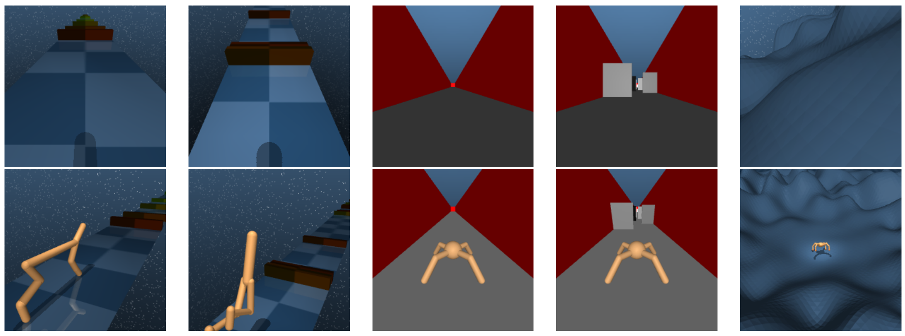


Figure 7.3.: The environments in the Locomotion Suite are (from left to right) `Hurdle Cheetah Run`, `Hurdle Walker Walk / Run`, `Ant Empty`, `Ant Walls`, and `Quadruped Escape`. **Upper Row:** Egocentric vision provided to the agent. **Lower Row:** External image for visualization.

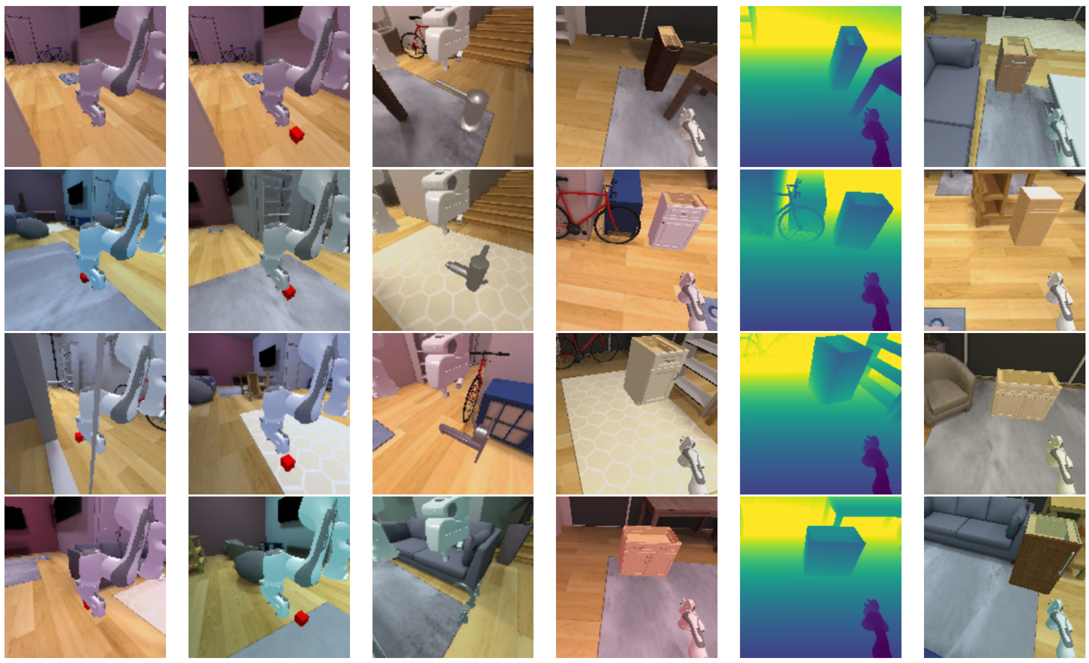


Figure 7.4.: 4 Example images for each of the environments in the Manipulation Suite, showing the visual and geometric diversity within each task. The tasks are, from left to right, `LiftCube`, `PushCube`, `TurnFaucet`, `OpenCabinetDrawer(RGB)`, `OpenCabinetDrawer(Depth)`, `OpenCabinetDoor(RGBD)`. For the last, we visualize only the RGB part of the image.

The `Quadruped Escape` task is readily available in the DeepMind Control Suite. For the egocentric vision, we removed the range-finding sensors from the original observation and added an egocentric camera.

7.4.4. Manipulation Suite

For the Manipulation suite, we design 6 tasks based on ManiSkill2 (Gu et al., 2023). Those tasks are `LiftCube`, `PushCube`, `TurnFaucet`, `OpenCabinetDrawer(RGB)`, `OpenCanbinetDrawer(Depth)`, and `OpenCabinetDoor(RGBD)`. The first three are static manipulation tasks where the target object has to be localized (cube) or identified (faucet) for successful manipulation. The latter three are mobile manipulation tasks where the robot navigates to a cabinet and interacts with it using egocentric vision and proprioception. They also use different visual modalities, i.e., standard RGB images, depth only, or RGBD. For all tasks, we add visually realistic backgrounds using diverse scenes from the ReplicaCAD Dataset (Straub et al., 2019)[1] and randomize the ambient lighting. At the beginning of each episode, we randomly pick one of 80 curated scenes and randomly sample the ambient lighting to place the task in a varying and visually realistic scenery. The task's complexity stems from the visual realism of the background and the diverse geometry of the target objects, which require that the representations allow identification and precise localization. Figure 7.4 provides example images showing the tasks' visual diversity. `LiftCube` builds on Maniskill2's LiftCube task and involves picking up a cube and lifting it to a fixed target position. The proprioception includes the robot's joint positions, velocities, and end-effector pose, while the cube has to be localized and tracked via an image of an external camera. Opposed to Zhu et al. (2023) we randomize the initial cube position, requiring the agents to first localize the cube based on the representation, which makes the task considerably more difficult.

`PushCube` builds on the PushCube task introduced by Zhu et al. (2023), but we again randomize the initial cube position. Like in LiftCube, the proprioception includes the robot's joint positions, velocities, and end-effector pose, while the cube has to be localized and tracked via an image of an external camera.

`TurnFaucet` extends Maniskill2's TurnFaucet task and involves opening various faucets by turning the handle. The proprioception includes the robot's joint positions, velocities, and end-effector pose, while all information regarding the faucet has to be inferred from an image of an external camera. We sample one out of 15 different faucets at the beginning of each episode. As their geometry and opening mechanism vary any representation needs to capture detailed information about the faucet and allow the policy to identify it. This makes our task considerably more difficult than that proposed by Zhu et al. (2023), who use the same faucet model for all episodes.

`OpenCabinetDrawer(RGB)` is based on the mobile manipulation OpenCabinetDrawer task from ManiSkill2, where a mobile robot with a single arm has to navigate towards and then open a drawer one of 25 cabinets. We disable the rotation of the robot base, which prevents the robot from turning away from the cabinet during initial exploration and significantly speeds up learning for all considered approaches. This results in a 10 dimensional action space, consisting of the x and y velocities of the base, desired changes for the 7 robot joints, and the gripper. Images are egocentric from the top of the robot base and the proprioception includes the entries from the ManiSkill2 "state dict".

`OpenCabinetDrawer(Depth)` is equivalent to `OpenCabientDrawer(RGB)` but the agent only receives an egocentric depth image instead of a color image. This effectively removes the variation in lighting from the environment.

For `OpenCabinetDoor(RGBD)` we build on the Maniskill2 task of the same name, use 25 different cabinet models, and the same action space as for `OpenCabientDrawer(RGB`. The sensory observations are also equivalent to the Drawer tasks, but we provide both color and depth information. While conceptually similar to the Drawer tasks opening the Door is considerably harder, as it requires coordination with the base not just to reach the handle, but also to pull back on it.

[1] specifically: ReplicaCAD_baked_lighting `https://huggingface.co/datasets/ai-habitat/ReplicaCAD_baked_lighting/`

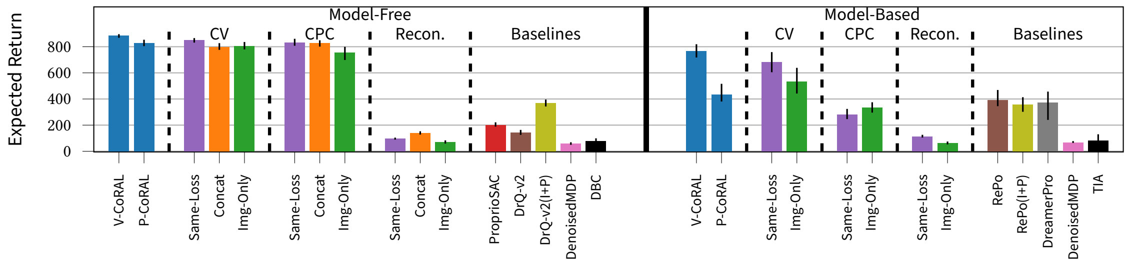


Figure 7.5.: Aggregated performance after 10^6 environment steps on the 7 tasks from the Video Background suite (IQM and 95% CIs). For both model-free and model-based RL, V-CoRAL performs best among all considered methods, with the model-free performance being better than the model-based one. While some of the model-free ablations are competitive, they perform considerably worse in the model-based case. From the baselines, only DrQ-v2 with additional proprioception, RePo (with and without proprioception), and DreamerPro get a final return of over 200. These results demonstrate how including readily available proprioception with appropriate losses for each modality helps to learn accurate dynamics required by model-based RL and provides a simple alternative to more tailored approaches.

7.5. Discussion of Results

We run 5 seeds for each task in each suite and build our analysis on the aggregated results across the entire suite. This process results in 35 runs for each method on Video Background and Occlusions and 30 runs for each method in the Locomotion and Manipulation suites. For aggregating the results over a suite, we follow Agarwal et al. (2021) and provide Interquartile Means (IQMs), which they found to be more meaningful and robust than alternatives such as mean or median in related scenarios. Similarly, we follow Agarwal et al. (2021) and provide 95% Stratified Bootstrapped Confidence Intervals (CIs) for the entire suite to quantify the statistical uncertainty in results. We indicate those with black bars in bar charts or shaded areas in reward curve plots.

Appendix E.1 lists all hyperparameters of our approach and Appendix E.2 provides further details on the baselines. Appendix E.3 shows learning curves for all representation learning paradigms on all tasks, performance profiles, and per-environment results.

7.5.1. Modified Deep Mind Control Suite Tasks

For the modified Deep Mind Control Suite tasks, we consider the model-free and model-based versions of V-CoRAL, P-CoRAL, and all ablative variants. Note that the Concat ablations are inapplicable in the model-based setting, as the proprioception is not available during latent imagination (Hafner et al., 2020). Additionally, we include several baselines, besides DrQ-v2 and RePo. Those are the model-based Task Informed Abstractions (TIA) (Fu et al., 2021) and DreamerPro (Deng et al., 2022), the model-free Deep Bisimulation for Control (DBC) (Zhang et al., 2020) approach, and DenoisedMDP (Wang et al., 2022), which has both a model-free and model-based variant. All these methods are visual RL approaches tailored for images with distractions and the comparison further shows the competitiveness of CoRAL.

Figure 7.5 and Figure 7.6 show the results for the Video Background and Occlusion tasks respectively. We also include results for the Standard Images without any distractors or occlusions for reference and refer to Appendix E.3 for those results. On Natural Videos V-CoRAL yields the best results

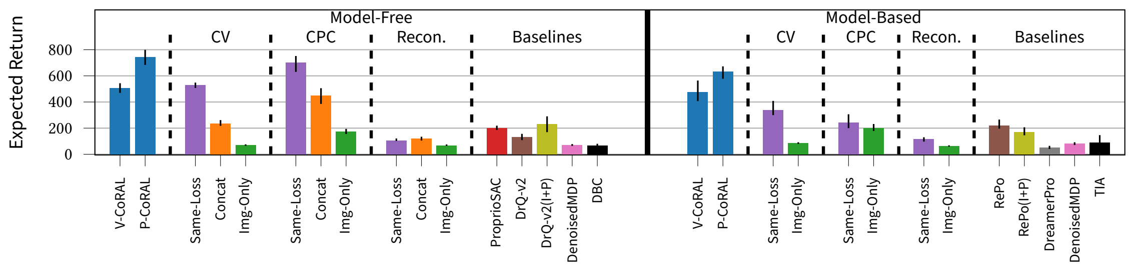


Figure 7.6.: Aggregated performance after 10^6 environment steps on the 7 tasks from the Occlusion suite (IQM and 95% CIs). For both, model-free and model-based RL, P-CoRAL performs best among all considered methods, with the model-free version again outperforming its model-based counterpart. While all approaches handle Occlusions worse than VideoBackgorund, the performance drop is generally larger for the ablations and baselines. In particular, the Concat and model-based Same-Loss ablations suffer and no approach using only a single modality achieves an expected return of over 200. This indicates the importance of learning a multi-sensor representation using tailored losses over naively integrating proprioception.

among all approaches. However, the margin to some of the ablations is small with all of them closing in on the performance of the best approaches on images without background videos (Figure E.1). For model-based RL the results show clearer benefits of learning a multi-sensor representation by appropriately combining multiple losses. This difference is also much more pronounced for the more difficult Occlusions (Figure 7.6) suite. Here, no image-only approach learns reasonable behavior or manages to outperform ProprioSAC, indicating a higher difficulty for representation learning. Our method P-CoRAL tends to perform best in this suite, achieving a return of around 750, and closing in on the best approaches on Standard Images which get around 900. Furthermore, using readily available proprioception for representation learning in a principled manner provides a simple alternative to the strong baselines listed above and also tends to outperform the naive Concat ablation that does not consider proprioception for representation learning but only for RL.

Variational vs. Predictive Approaches. Variational approaches tend to work better than predictive ones on Video Backgrounds, where the challenge is to focus on the relevant aspects while ignoring distractions. Yet, the predictive approaches work better on Occlusions, where information has to be propagated over time. As the underlying tasks are identical, this highlights the benefits of considering both paradigms, depending on the perception challenges.

Visualization of Learned Representations. We qualitatively investigate some of the learned representations in Figure 7.7, which illustrates how CoRAL helps the representation to focus on relevant aspects and extract all necessary information from an image.

Model Quality and Model-Based Approaches. While model-free and model-based agents perform similarly well for approaches that reconstruct images, model-based agents perform worse than their model-free counterparts for contrastive image losses (Figure 7.5, Figure 7.6, Figure E.1, Figure E.2). In line with previous findings (Hafner et al., 2020; Ma et al., 2020), this shows how contrastive approaches struggle to learn suitable long-term dynamics for model-based RL. However, this gap is larger for the Same-Loss and Img-Only ablations than for CoRAL, which almost closes the gap between model-free and model-based for V-CoRAL (Figure 7.5, Figure 7.6). This result

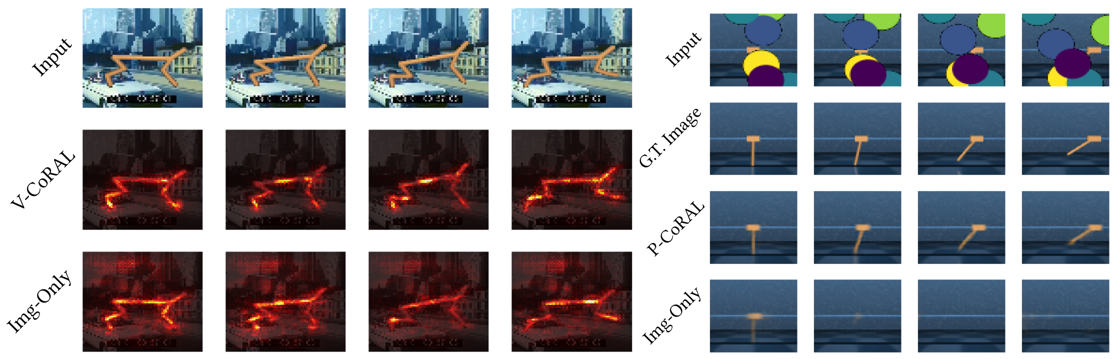


Figure 7.7.: Left: Saliency Maps showing on which pixels the respective representation learning approaches focus in an example from Video Prediction. V-CoRAL focuses better on the task-relevant cheetah, while the corresponding contrastive variational Img-Only approach is more distracted by the video background. **Right:** For this Occlusion task, we train a separate decoder to reconstruct the occlusion-free ground truth from the (detached) latent representation. For `Cartpole Swingup` only the cart position is part of the proprioception. Still, P-CoRAL can capture both cart position and pole angle, while the contrastive predictive Img-Only approach fails to do so.

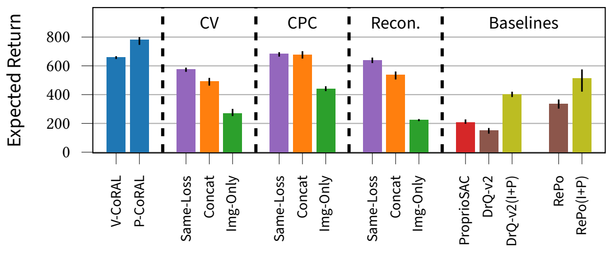


Figure 7.8.: Aggregated performance on model-free agents and RePo after 10^6 environment steps on the 6 tasks of the Locomotion suite (IQM and 95% CIs). P-CoRAL significantly outperforms all ablative variants and baselines, highlighting how combining contrastive methods and reconstruction can form effective multi-sensor representations. It also outperforms purely reconstruction-based approaches, even with no distraction in the images.

demonstrates how CoRAL allows learning more precise long-term dynamics that enable more successful model-based RL.

7.5.2. Locomotion Suite

For this more challenging Locomotion suite, we focus on model-free RL for all representations due to the performance gap for model-based RL with contrastive image losses (Hafner et al., 2020; Ma et al., 2020), (Figure 7.5, Figure 7.6). We include the model-based RePo for reference.

The results in Figure 7.8 show that P-CoRAL excels in the Locomotion suite and has a significant edge over reconstruction or the pure CPC-based approach while V-CoRAL outperforms the related variational approaches. While highly relevant to the task, the obstacles appear at random and have random colors for some tasks, which makes reconstruction harder. The contrastive methods' advantage is pronounced in tasks with random colored obstacles (Figure E.13).

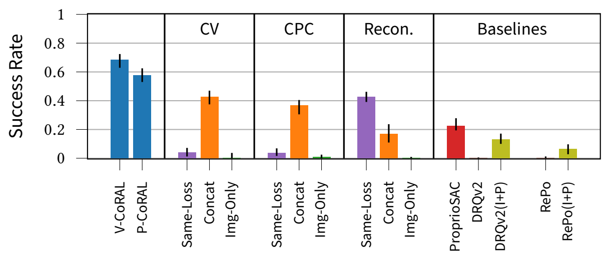


Figure 7.9.: Aggregated performance on model-free agents and RePo after 2×10^6 environment steps on the Manipulation suite (IQM and 95% CIs). Overall, V-CoRAL achieves the best average success rate by a significant margin, followed by P-CoRAL. While they achieve about 68% and 58% success respectively, no ablation gets over 42%. In particular, both fully contrastive Same-Loss ablations fail to succeed, which again highlights the importance of choosing an appropriate loss for each modality. While both RePo and DrQ-v2 can utilize the additional proprioception, they are not competitive with CoRAL or even SAC trained solely on the proprioception.

7.5.3. Manipulation Suite

For our most challenging set of tasks, the Manipulation suite, we again focus on model-free RL and RePo. Figure 7.9 shows the results. The Manipulation suite is the hardest set of tasks we consider and here the benefits of CoRAL are most obvious. Here, most of the considered baselines fail while only V-CoRAL and P-CoRAL achieve over 50% success rate, averaged over all tasks, with V-CoRAL giving the best result of 68%. In particular, the corresponding contrastive same-loss approaches fail almost completely, which puts additional emphasis on the importance of using appropriate losses for each modality. Using different image types for the 3 mobile manipulation tasks shows how CoRAL is beneficial across different visual modalities. Using depth images, `OpenCabinetDrawer(Depth)` effectively removes the lighting variations for this task which allows several approaches to achieve higher performance but has only minor effects on the ranking.

7.5.4. Discussion

Considering all task suites and the full results presented in Appendix E.3, we see the benefits of CoRAL compared to the ablations and a large selection of model-free and model-based baselines. Especially for the harder tasks, i.e., Occlusions (Figure 7.6), Locomotion (Figure 7.8), and Manipulation (Figure 7.9), CoRAL can significantly outperform other methods working on the same observations, which shows that different modalities require distinct self-supervised loss functions while simply using the additional proprioception by concatenation or using the same self-supervised loss is often insufficient.

Variational vs. Predictive Approaches. While either V-CoRAL or P-CoRAL generally provides the best results on the considered tasks and both outperform the corresponding ablations, neither consistently outperforms the other across all task suites. While this prevents conclusive decisions about whether variational or predictive methods work generally better, we can observe a trend. The variational approaches appear more suited for tasks requiring the filtering out of visual distractions, such as in Video Background and Manipulation scenarios, while predictive approaches perform better in tasks needing information to be carried over time, like Occlusions and Locomotion.

Baselines. Contrary to the results presented in the respective original works, DBC (Zhang et al., 2020), TIA (Fu et al., 2021), DenoisedMDP (Wang et al., 2022) and RePo (Zhu et al., 2023) underperform on the Video Backgrounds task. The discrepancy in performance is due to us using a more difficult experimental setup proposed by Deng et al. (2022), which features colored videos of greater diversity. We detail the differences and their effects in Appendix E.2. Furthermore, RePo fails in the Manipulation suite which seems to contradict results presented by Zhu et al. (2023) on three static manipulation tasks, similar to those in that suite. Again there are subtle differences in the task specification: While Zhu et al. (2023) only randomize the visual background we randomize both the visual background and the task's initial condition (cube position or faucet model) creating considerably more challenging scenarios.

Consistency Across Tasks. The additional result visualizations in Appendix E.3 show that the aggregated performance underlying our analysis is mostly representative of the per-task performance, i.e., if an approach outperforms another when considering the aggregated performance, it generally also does so on a large majority of the individual tasks and runs. Furthermore, performance is consistent across the different observation types for the DMC tasks, i.e., Occlusions are more difficult than Video Background, which are more difficult than Standard Images (Figure E.1, Figure E.2).

7.6. Conclusion

We consider the problem of Reinforcement Learning (RL) from multiple sensors, in particular images and proprioception. We propose Contrastive reconstructive Aggregated Representation Learning (CoRAL), an approach to learning multi-sensor state space representations for RL by combining contrastive and reconstruction losses. CoRAL builds on the insight that we can replace likelihood-based reconstruction terms with contrastive mutual information terms and vice-versa and is applicable for variational and predictive coding paradigms. We evaluate on modified versions of the DeepMind Control Suite and novel Locomotion and Manipulation suites. Our results show a consistent benefit of CoRAL due to the combination of contrastive approaches for images with reconstruction for low-dimensional, concise signals. These benefits are most pronounced for the hardest tasks we consider, i.e., the Manipulation suite, where CoRAL, allows us to solve complex tasks with realistic background scenes and varying target object geometries.

Limitations. Depending on the task, either V-CoRAL or P-CoRAL performs better. While our evaluation provides some insights about when to use either, further research into understanding their advantages and disadvantages and finding a unified approach that excels in all tasks is required. Additionally, even with CoRAL, model-free agents outperform their model-based counterparts when using contrastive image losses. We thus believe that contrastive learning of state space representations can be further improved, especially with regard to learning accurate system dynamics.

Contribution Statement. *I devised and derived the CoRAL model, implemented it, devised and ran all evaluations, and wrote the large majority of the article. SM conducted preliminary evaluated different representation learning methods for image-only RL during his master's thesis under my supervision. Choosing the Mutual Information as contrastive loss is based on this analysis and I reused* *parts of his code for both the algorithm and the evaluation. He also helped with writing the article. FO helped with conduction evaluations and writing the article, in particular focusing on the presentation of the results. GN supervised the project.*

8. Conclusion

This thesis addresses the challenge of enabling embodied agents to operate safely and efficiently in complex and challenging environments. To act intelligently, these agents must not only perceive their surroundings but also understand the consequences of their actions and be able to reason about and imitate human behavior. Under real-world conditions, this requires them to deal with inherent unpredictabilities and uncertainties in both observations and dynamics, as well as multimodal human behavior. Thus, simple deterministic models, which assume a single correct state and a single outcome for each action, are brittle in the face of such ambiguity and uncertainty and can lead to inefficient or even dangerous behavior. Consequently, we build on the premise that robust, probabilistic world and behavior models are crucial for Imitation Learning (IL) and Reinforcement Learning (RL) with embodied agents.

To guide our research into using such probabilistic models to advance embodied agents, we formulated three research questions, first posed in Chapter 1:

Q1 How can we learn multimodal probabilistic models that focus on modes they can represent and avoid averaging? (Chapter 3)

Q2 How can we build a computationally efficient world model for RL that captures and propagates uncertainties in the world while separating them from uncertainties due to lack of training data? (Chapter 5 and Chapter 6)

Q3 How can we use probabilistic world models to combine information from multiple sensors with highly different properties into a single, concise representation as a basis for RL? (Chapter 7)

In answering these questions, we not only advance the capabilities of embodied agents but also focus on developing, analyzing, and improving probabilistic models as a basis for decision-making in complex environments. We introduce novel algorithms and architectures that leverage structured latent variable models to learn meaningful representations from multimodal data as well as high-dimensional, noisy, and partial observations. Specifically, our algorithmic contributions include probabilistic modelling itself and proposing novel learning objectives and inference schemes. For learning meaningful representations, we introduce new training methods tailored to avoid averaging over modes when learning from multimodal data, as well as a unified framework for combining reconstruction and contrastive losses to learn efficient representations from heterogeneous sensor data. For inference, we analyze the limitations of existing deep State Space Models (SSMs) and propose a principled approach to uncertainty modeling that disentangles different sources of uncertainty. Here, our improved smoothing inference scheme builds on Kalman smoothing in a latent space. Additionally, we propose novel architectures for deep probabilistic State Space Models (SSMs) that facilitate our improved inference scheme and enable efficient parallelization over the time axis, thus improving computational efficiency. Finally, we ensure that our models are not only good at learning representations and uncertainties but also amenable to decision-making through IL and RL.

The research presented in this thesis builds on a series of individual works. The following summarizes the key contributions of each of these works.

Chapter 3: Expected Information Maximization (EIM). Standard IL methods based on Maximum Likelihood (the M-Projection) tend to average over different modes of expert behavior, which can lead to catastrophic failures, such as an agent attempting to navigate through the center of an obstacle when demonstrated paths go around both sides. To overcome this, we introduce Expected Information Maximization (EIM), a novel algorithm for fitting mixture models using the Information Projection (I-Projection). Unlike the M-Projection, the I-Projection is mode-seeking, allowing the model to focus its capacity on representing a subset of the demonstrated behaviors accurately rather than averaging them. We derive a practical, sample-based algorithm for optimizing this objective, which is related to adversarial approaches (Goodfellow et al., 2014; Ho and Ermon, 2016) without being itself adversarial. Through experiments involving multimodal behavior, such as that of pedestrians and traffic, we demonstrate that EIM effectively learns multimodal distributions that produce more realistic and safer behaviors.

Chapter 5: On Uncertainty in Deep State Space Models (VRKN). We begin with an in-depth analysis of Recurrent State Space Models (RSSMs) (Hafner et al., 2019), a prominent backbone of many successful model-based RL agents. Here, our analysis reveals that the simplified filtering-based inference scheme used by RSSMs leads to a systematic overestimation of aleatoric uncertainty. While this overestimation acts as an implicit form of regularization that is beneficial in deterministic environments, it impairs performance in settings where accurately modeling uncertainty is crucial. To address this, we propose the Variational Recurrent Kalman Network (VRKN), a more principled deep SSM that combines a proper smoothing inference scheme based on Kalman updates (Kalman, 1960; Rauch et al., 1965) with an explicit model of epistemic uncertainty using Monte Carlo Dropout (Gal and Ghahramani, 2016). The VRKN's ability to perform exact inference in its latent space enables it to correctly disentangle aleatoric and epistemic uncertainties. Our experiments demonstrate that the VRKN matches the performance of RSSMs on standard benchmarks while significantly outperforming them in tasks involving occlusions and missing observations, highlighting the benefits of a principled approach to uncertainty modeling.

Chapter 6: KalMamba. While principled, the VRKN's recurrent nature prevents efficient parallelization over the time axis, limiting its scalability to long interaction sequences. We introduce KalMamba to bridge the gap between the probabilistic rigor of models like the VRKN and the computational efficiency of modern deterministic structured SSMs (Gu et al., 2021; Smith et al., 2022; Gu and Dao, 2023). This architecture leverages a Mamba (Gu and Dao, 2023) backbone to learn the parameters of a Linear Gaussian SSM in a latent space. It then utilizes a time-parallel formulation of Kalman filtering and smoothing (Särkkä and García-Fernández, 2020) operations for inference given the learned model. Hence, KalMamba is a highly efficient and scalable deep probabilistic SSM that retains the benefits of a tight variational bound from smoothing inference while leveraging modern accelerators. Our experiments demonstrate that KalMamba is competitive with state-of-the-art probabilistic SSMs in RL tasks while offering significant improvements in computational efficiency, especially for long sequences.

Chapter 7: Combining Reconstruction and Contrastive Methods (CoRAL). We argue that a one-size-fits-all approach to representation learning is suboptimal when learning in settings with multiple sensor modalities which have different properties. For instance, a reconstruction-based loss is well-suited for clean, low-dimensional proprioceptive data but can be distracted by irrelevant information in high-dimensional images. Conversely, contrastive losses can ignore such distractions but may fail to capture fine-grained details (Hafner et al., 2020; Deng et al., 2022) Thus, we propose Contrastive Reconstructive Aggregated representation Learning (CoRAL), a unified framework that allows the flexible combination of reconstruction and contrastive losses for each sensor modality. By exploiting the interchangeability of likelihood and mutual information objectives within both variational and predictive coding (Oord et al., 2018) paradigms, CoRAL enables the learning of a single, concise state space representation tailored to the specific characteristics of the available sensors. Across a wide range of challenging tasks involving visual distractions, occlusions, and multi-sensor fusion, we demonstrate that CoRAL significantly improves the performance of both model-free and model-based agents, highlighting the benefits of adapting the learning objective to the nature of the sensory input.

In summary, the contributions of this thesis provide a coherent set of advancements in probabilistic modeling for embodied agents. While we address several specific research questions, they collectively form a foundation upon which future research can build and open up new and exciting directions for creating more intelligent and capable embodied agents.

8.1. Future Work and Outlook

While this thesis has advanced the development of probabilistic world and behavior models for embodied agents, we are far from done, and many exciting challenges remain. To move closer to the goal of building truly autonomous agents that can solve a variety of tasks in the real world, we must continue to improve and scale the kind of probabilistic models developed in this work. In this context, the recent successes of large-scale generative AI and foundation models are already causing a paradigm shift (Firoozi et al., 2025; Gemini Robotics Team, 2025). These models have demonstrated remarkable capabilities and provide powerful, pre-trained priors for perception that generalize well across large domains (Kim et al., 2024). Furthermore, they make new modalities, such as language, available to embodied agents, enabling them to reason and communicate with humans more naturally. However, while some works aim to equip them with principled notions of uncertainty, such as through Bayesian fine-tuning (Seligmann et al., 2023; Pandey et al., 2024), the features extracted by these models are usually deterministic, and it is unclear whether or how they represent uncertainty. Thus, naively using them as feature extractors in our probabilistic state space framework might be insufficient, and future work must explore how to leverage the internal representations of foundation models and make their remarkable capabilities amenable to probabilistic reasoning.

Furthermore, all agents developed in this work are specialists, trained to solve a single task. However, truly autonomous and general embodied agents must be able to solve a wide range of tasks and adapt to new ones without extensive retraining. In the fully observable Markov Decision Process (MDP) setting, training methods for such agents are currently developed using paradigms such as meta RL (Rakelly et al., 2019; Zintgraf et al., 2020), goal-conditioned (Eysenbach et al., 2022) or multi-task RL, or unsupervised RL (Eysenbach et al., 2019; Tirinzoni et al., 2025) Yet, regardless of the paradigm, these works typically assume fully observable, concise, and unrealistically simple state representations, allowing for a complete focus on generalizing across tasks. Naively combining these approaches with one of our, or potential future, probabilistic SSMs might be a first step towards more general embodied agents. However, especially when dealing with the types of highly challenging observations considered in this thesis, it is likely beneficial to learn a task or goal-aware world model that can focus on the relevant aspects of the environment for the task at hand. How to design such a model in a principled way, particularly incorporating the benefits from the previously mentioned foundation models, is an open question that requires further research.

In the context of IL, this trend towards more general agents is currently already visible in the form of modern diffusion-based policies (Chi et al., 2023). These methods have shown remarkable success in learning complex behaviors from large, diverse datasets conditioned on language prompts or images of the desired behavior. While these methods are vastly more capable than the older Expected Information Maximization (EIM) approach presented in this thesis, they again employ the maximum likelihood objective or related score-matching objectives (Song et al., 2021) and rely on the model being sufficiently rich to capture multimodal behaviors without averaging. However, as seen in Chapter 3, using the I-Projection can still offer benefits even when the model is sufficiently rich to capture multimodal expert behavior. Furthermore, while we focused on Mixture Models, EIM is a much more general framework applicable to arbitrary latent variable models, including diffusion models. Thus, investigating how to use insights from EIM to train large-scale diffusion policies could be a promising direction for future work to improve safety and adherence to expert behavior.

In summary, I believe that the path toward more capable and general embodied agents lies in combining principled probabilistic models and objectives with the remarkable capabilities of large-scale foundation models. While the community currently focuses mainly on scaling up foundation models and improving their capabilities, I believe that the simultaneous and continued development of scalable yet principled probabilistic models and inference schemes is essential for the next generation of truly autonomous general and embodied agents.

Bibliography

P. Abbeel and A. Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 1, 2004.

A. Abdolmaleki, R. Lioutikov, J. R. Peters, N. Lau, L. P. Reis, and G. Neumann. Model-based relative entropy stochastic search. In *Advances in Neural Information Processing Systems*, pages 3537–3545, 2015.

R. Agarwal, M. Schwarzer, P. S. Castro, A. C. Courville, and M. Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in neural information processing systems*, 34:29304–29320, 2021.

A. A. Alemi, I. Fischer, J. V. Dillon, and K. Murphy. Deep variational information bottleneck. In *International Conference on Learning Representations*, 2016.

S. M. Ali and S. D. Silvey. A general class of coefficients of divergence of one distribution from another. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 131–142, 1966.

B. Amos and J. Z. Kolter. Optnet: Differentiable optimization as a layer in neural networks. In *International conference on machine learning*, pages 136–145. PMLR, 2017.

B. D. Anderson and J. B. Moore. *Optimal control: linear quadratic methods.* Courier Corporation, 2007.

E. Archer, I. M. Park, L. Buesing, J. Cunningham, and L. Paninski. Black box variational inference for state space models. *arXiv preprint arXiv:1511.07367*, 2015.

O. Arenz, M. Zhong, and G. Neumann. Efficient gradient-free variational inference using policy search. In *Proceedings of the 35th International Conference on Machine Learning*, pages 234–243. pmlr, 2018.

O. Arenz, M. Zhong, and G. Neumann. Trust-region variational inference with gaussian mixture models. *Journal of Machine Learning Research*, 21(163):1–60, 2020.

O. Arenz, P. Dahlinger, Z. Ye, M. Volpp, and G. Neumann. A unified perspective on natural gradient variational inference with gaussian mixture models. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856.

M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Transactions on signal processing*, 50(2):174–188, 2002.

K. J. Astrom. Optimal control of markov decision processes with incomplete state estimation. *J. Math. Anal. Applic.*, 10:174–205, 1965.

J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

A. P. Badia, B. Piot, S. Kapturowski, P. Sprechmann, A. Vitvitskyi, Z. D. Guo, and C. Blundell. Agent57: Outperforming the atari human benchmark. In *International conference on machine learning*, pages 507–517. PMLR, 2020.

E. Banijamali, R. Shu, H. Bui, A. Ghodsi, et al. Robust locally-linear controllable embedding. In *International Conference on Artificial Intelligence and Statistics*, pages 1751–1759. PMLR, 2018.

P. Becker and G. Neumann. On uncertainty in deep state space models for model-based reinforcement learning. *Transactions on Machine Learning Research*, 2022.

P. Becker, H. Pandya, G. Gebhardt, C. Zhao, C. J. Taylor, and G. Neumann. Recurrent kalman networks: Factorized inference in high-dimensional deep feature spaces. In *International Conference on Machine Learning*, pages 544–552, 2019.

P. Becker, O. Arenz, and G. Neumann. Expected information maximization: Using the i-projection for mixture density estimation. In *International Conference on Learning Representations*, 2020.

P. Becker, N. Freymuth, and G. Neumann. Kalmamba: Towards efficient probabilistic state space models for rl under uncertainty. In *ICML 2024 Workshop: Aligning Reinforcement Learning Experimentalists and Theorists*, 2024a.

P. Becker, S. Mossburger, F. Otto, and G. Neumann. Combining reconstruction and contrastive methods for multimodal representations in RL. *Reinforcement Learning Journal*, 4:1619–1655, 2024b.

P. Becker-Ehmck, J. Peters, and P. Van Der Smagt. Switching linear dynamics for variational Bayes filtering. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 553–562. PMLR, 09–15 Jun 2019.

R. Bellman. A markovian decision process. *Journal of mathematics and mechanics*, pages 679–684, 1957.

C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006. ISBN 0387310738.

G. E. Blelloch. Prefix sums and their applications, 1990.

L. M. Bregman. The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR computational mathematics and mathematical physics*, 7(3):200–217, 1967.

A. Brohan, Y. Chebotar, C. Finn, K. Hausman, A. Herzog, D. Ho, J. Ibarz, A. Irpan, E. Jang, R. Julian, et al. Do as i can, not as i say: Grounding language in robotic affordances. In *6th Annual Conference on Robot Learning*, 2022.

D. Büchler, H. Ott, and J. Peters. A lightweight robotic arm with pneumatic muscles for robot learning. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4086–4092. IEEE, 2016.

L. Buesing, T. Weber, S. Racaniere, S. Eslami, D. Rezende, D. P. Reichert, F. Viola, F. Besse, K. Gregor, D. Hassabis, et al. Learning and querying fast generative models for reinforcement learning. *arXiv preprint arXiv:1802.03006*, 2018.

O. Celik, A. Taranovic, and G. Neumann. Acquiring diverse skills using curriculum reinforcement learning with mixture of experts. In *Forty-first International Conference on Machine Learning*, 2024.

C. Chen, J. Yoon, Y.-F. Wu, and S. Ahn. Transdreamer: Reinforcement learning with transformer world models, 2021.

L. Chen, S. Dai, Y. Pu, E. Zhou, C. Li, Q. Su, C. Chen, and L. Carin. Symmetric variational autoencoder and connections to adversarial learning. In *International Conference on Artificial Intelligence and Statistics*, pages 661–669, 2018.

C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, page 02783649241273668, 2023.

K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

K. Chua, R. Calandra, R. McAllister, and S. Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *Advances in Neural Information Processing Systems*, 31, 2018.

D.-A. Clevert, T. Unterthiner, and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

J. Degrave, F. Felici, J. Buchli, M. Neunert, B. Tracey, F. Carpanese, T. Ewalds, R. Hafner, A. Abdolmaleki, D. de Las Casas, et al. Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature*, 602(7897):414–419, 2022.

M. Deisenroth and C. E. Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pages 465–472. Citeseer, 2011.

M. P. Deisenroth, G. Neumann, J. Peters, et al. A survey on policy search for robotics. *Foundations and Trends® in Robotics*, 2(1–2):1–142, 2013.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977.

F. Deng, I. Jang, and S. Ahn. Dreamerpro: Reconstruction-free model-based reinforcement learning with prototypical representations. In *International Conference on Machine Learning*, pages 4956–4975. PMLR, 2022.

A. Doerr, C. Daniel, M. Schiegg, N.-T. Duy, S. Schaal, M. Toussaint, and T. Sebastian. Probabilistic recurrent state-space models. In *International Conference on Machine Learning*, pages 1280–1289. PMLR, 2018.

D. Driess, F. Xia, M. S. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter, A. Wahid, J. Tompson, Q. Vuong, T. Yu, et al. Palm-e: An embodied multimodal language model. In *International Conference on Machine Learning*. PMLR, 2023.

S. Eleftheriadis, T. Nicholson, M. P. Deisenroth, and J. Hensman. Identification of gaussian process state space models. In *NIPS*, pages 5309–5319, 2017.

J. L. Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

L. Engstrom, A. Ilyas, S. Santurkar, D. Tsipras, F. Janoos, L. Rudolph, and A. Madry. Implementation matters in deep rl: A case study on ppo and trpo. In *International conference on learning representations*, 2019.

B. Eysenbach, J. Ibarz, A. Gupta, and S. Levine. Diversity is all you need: Learning skills without a reward function. In *7th International Conference on Learning Representations, ICLR 2019*, 2019.

B. Eysenbach, T. Zhang, S. Levine, and R. R. Salakhutdinov. Contrastive learning as goal-conditioned reinforcement learning. *Advances in Neural Information Processing Systems*, 35:35603–35620, 2022.

C. Finn, I. Goodfellow, and S. Levine. Unsupervised learning for physical interaction through video prediction. *Advances in neural information processing systems*, 29, 2016.

R. Firoozi, J. Tucker, S. Tian, A. Majumdar, J. Sun, W. Liu, Y. Zhu, S. Song, A. Kapoor, K. Hausman, et al. Foundation models in robotics: Applications, challenges, and the future. *The International Journal of Robotics Research*, 44(5):701–739, 2025.

M. Fraccaro, S. Kamronn, U. Paquet, and O. Winther. A disentangled recognition and nonlinear dynamics model for unsupervised learning. In *Advances in Neural Information Processing Systems*, pages 3601–3610, 2017.

N. Freymuth, P. Becker, and G. Neumann. Versatile inverse reinforcement learning via cumulative rewards. In *NeurIPS 2021 Workshop on Robot Learning: Self-Supervised and Lifelong Learning, Virtual*, 2021.

N. Freymuth, N. Schreiber, A. Taranovic, P. Becker, and G. Neumann. Inferring versatile behavior from demonstrations by matching geometric descriptors. In *6th Annual Conference on Robot Learning*, 2022.

X. Fu, G. Yang, P. Agrawal, and T. Jaakkola. Learning task informed abstractions. In *International Conference on Machine Learning*, pages 3480–3491. PMLR, 2021.

Z. Fu, A. Kumar, A. Agarwal, H. Qi, J. Malik, and D. Pathak. Coupling vision and proprioception for navigation of legged robots. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 17273–17283, June 2022.

Y. Gal and Z. Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR, 2016.

Y. Gal and C. E. Rasmussen. Improving pilco with bayesian neural network dynamics models. In *In Data-efficient machine learning workshop, ICML*, 2016.

G. Gebhardt, A. Kupcsik, and G. Neumann. The kernel kalman rule - efficient nonparametric inference with recursive least squares. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

Gemini Robotics Team. Gemini robotics: Bringing ai into the physical world. *arXiv preprint arXiv:2503.20020*, 2025.

Z. Ghahramani and G. E. Hinton. Variational learning for switching state-space models. *Neural computation*, 12(4):831–864, 2000.

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

I. Goodfellow, Y. Bengio, and A. Courville. Deep learning, 2016.

E. Gospodinov, V. Shaj, P. Becker, S. Geyer, and G. Neumann. Adaptive world models: Learning behaviors by latent imagination under non-stationarity. In *Adaptive Foundation Models: Evolving AI for Personalized and Efficient Learning*, 2024.

A. Gu and T. Dao. Mamba: Linear-time sequence modeling with selective state spaces, 2023.

A. Gu, K. Goel, and C. Re. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations*, 2021.

J. Gu, F. Xiang, X. Li, Z. Ling, X. Liu, T. Mu, Y. Tang, S. Tao, X. Wei, Y. Yao, X. Yuan, P. Xie, Z. Huang, R. Chen, and H. Su. Maniskill2: A unified benchmark for generalizable manipulation skills. In *International Conference on Learning Representations*, 2023.

S. Gu, Z. Ghahramani, and R. Turner. Neural adaptive sequential monte carlo. *Advances in Neural Information Processing Systems*, 2015:2629–2637, 2015.

A. Gupta, A. Gu, and J. Berant. Diagonal state spaces are as effective as structured state spaces. *Advances in Neural Information Processing Systems*, 35:22982–22994, 2022.

F. Gustafsson. *Statistical sensor fusion*. Studentlitteratur, 2010.

D. Ha and J. Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

T. Haarnoja, A. Ajay, S. Levine, and P. Abbeel. Backprop kf: learning discriminative deterministic state estimators. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 4383–4391, 2016.

T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

D. Hafner, T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson. Learning latent dynamics for planning from pixels. In *International Conference on Machine Learning*, pages 2555–2565. PMLR, 2019.

D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2020.

D. Hafner, T. P. Lillicrap, M. Norouzi, and J. Ba. Mastering atari with discrete world models. In *International Conference on Learning Representations*, 2021.

D. Hafner, K.-H. Lee, I. Fischer, and P. Abbeel. Deep hierarchical planning from pixels. In *Advances in Neural Information Processing Systems*, 2022.

D. Hafner, J. Pasukonis, J. Ba, and T. Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

N. Hansen, H. Su, and X. Wang. Td-mpc2: Scalable, robust world models for continuous control. In *The Twelfth International Conference on Learning Representations*, 2024.

M. Harris, S. Sengupta, and J. D. Owens. Parallel prefix sum (scan) with cuda. *GPU gems*, 3(39): 851–876, 2007.

R. Hasani, M. Lechner, T.-H. Wang, M. Chahine, A. Amini, and D. Rus. Liquid structural state-space models. In *The Eleventh International Conference on Learning Representations*, 2022.

M. Hessel, J. Modayil, H. Van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

J.-B. Hiriart-Urruty and C. Lemaréchal. *Fundamentals of convex analysis.* Springer Science & Business Media, 2012.

J. Ho and S. Ermon. Generative adversarial imitation learning. *Advances in neural information processing systems*, 29, 2016.

T. Hoang, H. Le, P. Becker, V. A. Ngo, and G. Neumann. Geometry-aware RL for manipulation of varying shapes and deformable objects. In *The Thirteenth International Conference on Learning Representations*, 2025.

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

E. Hüllermeier and W. Waegeman. Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods. *Machine Learning*, 110(3):457–506, 2021.

R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87, 1991.

M. Janner, J. Fu, M. Zhang, and S. Levine. When to trust your model: Model-based policy optimization. *Advances in Neural Information Processing Systems*, 32:12519–12530, 2019.

A. Jazwinski. *Stochastic processes and filtering theory.* ACADEMIC PRESS, INC.„ 1970.

M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul. An introduction to variational methods for graphical models. *Machine learning*, 37:183–233, 1999.

S. J. Julier and J. K. Uhlmann. New extension of the kalman filter to nonlinear systems. In *Signal processing, sensor fusion, and target recognition VI*, volume 3068, pages 182–193. Spie, 1997.

L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998.

S. M. Kakade. A natural policy gradient. In *Advances in neural information processing systems*, pages 1531–1538, 2002.

D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, et al. Scalable deep reinforcement learning for vision-based robotic manipulation. In *Conference on Robot Learning*, pages 651–673. PMLR, 2018.

R. E. Kalman. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960.

M. Karl, M. Soelch, J. Bayer, and P. van der Smagt. Deep variational bayes filters: Unsupervised learning of state space models from raw data, 2016.

W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017.

B. Khaleghi, A. Khamis, F. O. Karray, and S. N. Razavi. Multisensor data fusion: A review of the state-of-the-art. *Information fusion*, 14(1):28–44, 2013.

R. Kidambi, A. Rajeswaran, P. Netrapalli, and T. Joachims. Morel: Model-based offline reinforcement learning. *Advances in neural information processing systems*, 33:21810–21823, 2020.

M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, P. Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

D. Kingma and J. Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014.

D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

A. Klushyn, R. Kurle, M. Soelch, B. Cseke, and P. van der Smagt. Latent matters: Learning deep state-space models. *Advances in Neural Information Processing Systems*, 34, 2021.

D. Koller and N. Friedman. *Probabilistic graphical models: principles and techniques.* MIT press, 2009.

R. Krishnan, U. Shalit, and D. Sontag. Structured inference networks for nonlinear state space models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

R. G. Krishnan, U. Shalit, and D. Sontag. Deep kalman filters. *arXiv preprint arXiv:1511.05121*, 2015.

S. Kullback and R. A. Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

T. Kurutach, I. Clavera, Y. Duan, A. Tamar, and P. Abbeel. Model-ensemble trust-region policy optimization. In *International Conference on Learning Representations*, 2018.

B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, 30, 2017.

N. Lambert, B. Amos, O. Yadan, and R. Calandra. Objective mismatch in model-based reinforcement learning. *Proceedings of Machine Learning Research vol*, 120:1–15, 2020.

M. Laskin, A. Srinivas, and P. Abbeel. CURL: Contrastive unsupervised representations for reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 5639–5650. PMLR, 13–18 Jul 2020.

A. X. Lee, A. Nagabandi, P. Abbeel, and S. Levine. Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model. *Advances in Neural Information Processing Systems*, 33: 741–752, 2020.

N. Levine, Y. Chow, R. Shu, A. Li, M. Ghavamzadeh, and H. Bui. Prediction, consistency, curvature: Representation learning for locally-linear control. In *International Conference on Learning Representations*, 2019.

S. Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*, 2018.

S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

C. Li, K. Bai, J. Li, G. Wang, C. Chen, and L. Carin. Adversarial learning of a sampler based on an unnormalized distribution. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 3302–3311, 2019.

Y. Luo, H. Xu, Y. Li, Y. Tian, T. Darrell, and T. Ma. Algorithmic framework for model-based deep reinforcement learning with theoretical guarantees. In *International Conference on Learning Representations*, 2019.

M. Lutter, L. Hasenclever, A. Byravan, G. Dulac-Arnold, P. Trochim, N. Heess, J. Merel, and Y. Tassa. Learning dynamics models for model predictive agents. *arXiv preprint arXiv:2109.14311*, 2021.

X. Ma, S. Chen, D. Hsu, and W. S. Lee. Contrastive variational reinforcement learning for complex observations. In *Conference on Robot Learning*, pages 959–972. PMLR, 2020.

L. Maaløe, C. K. Sønderby, S. K. Sønderby, and O. Winther. Auxiliary deep generative models. In *International Conference on Machine Learning*, pages 1445–1453, 2016.

D. Q. Mayne. Model predictive control: Recent developments and future promise. *Automatica*, 50 (12):2967–2986, 2014.

G. J. McLachlan and T. Krishnan. *The EM algorithm and extensions.* John Wiley & Sons, 2008.

O. Mees, L. Hermann, and W. Burgard. What matters in language conditioned robotic imitation learning over unstructured data. *IEEE Robotics and Automation Letters*, 7(4):11205–11212, 2022.

V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

T. M. Moerland, J. Broekens, A. Plaat, C. M. Jonker, et al. Model-based reinforcement learning: A survey. *Foundations and Trends® in Machine Learning*, 16(1):1–118, 2023.

A. Moretti, Z. Wang, L. Wu, and I. Pe'er. Smoothing nonlinear variational objectives with sequential monte carlo, 2019.

K. P. Murphy. Switching kalman filters, 1998.

K. P. Murphy. *Machine learning: a probabilistic perspective.* MIT press, 2012.

C. Naesseth, S. Linderman, R. Ranganath, and D. Blei. Variational sequential monte carlo. In *International Conference on Artificial Intelligence and Statistics*, pages 968–977. PMLR, 2018.

R. M. Neal and G. E. Hinton. A view of the em algorithm that justifies incremental, sparse, and other variants. In *Learning in graphical models*, pages 355–368. Springer, 1998.

A. Y. Ng and S. J. Russell. Algorithms for inverse reinforcement learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, page 663–670, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc. ISBN 1558607072.

T. D. Nguyen, R. Shu, T. Pham, H. Bui, and S. Ermon. Temporal predictive coding for model-based planning in latent space. In *International Conference on Machine Learning*, pages 8130–8139. PMLR, 2021.

X. Nguyen, M. J. Wainwright, and M. I. Jordan. Estimating divergence functionals and the likelihood ratio by convex risk minimization. *IEEE Transactions on Information Theory*, 56(11):5847–5861, 2010.

G. Nguyen-Quynh, P. Becker, C. Qiu, M. Rudolph, and G. Neumann. Switching recurrent kalman networks. *Machine Learning for Autonomous Driving Workshop at NeurIPS 2021*, 2021.

S. Nowozin, B. Cseke, and R. Tomioka. f-gan: Training generative neural samplers using variational divergence minimization. In *Advances in Neural Information Processing Systems*, pages 271–279, 2016.

M. O'Connell, M. Matias, and W. Kan. Ecml/pkdd 15: Taxi trajectory prediction (i), 2015. Kaggle.

M. Okada and T. Taniguchi. Dreaming: Model-based reinforcement learning by latent imagination without reconstruction. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4209–4215. IEEE, 2021.

M. Okada and T. Taniguchi. Dreamingv2: Reinforcement learning with discrete world models without reconstruction. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 985–991. IEEE, 2022.

M. Okada, N. Kosaka, and T. Taniguchi. Planet of the bayesians: Reconsidering and improving deep planning network by incorporating bayesian inference. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5611–5618. IEEE, 2020.

A. v. d. Oord, Y. Li, and O. Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

M. Opper and D. Saad. *Advanced mean field methods: Theory and practice.* MIT press, 2001.

T. Osa, J. Pajarinen, G. Neumann, J. A. Bagnell, P. Abbeel, J. Peters, et al. An algorithmic perspective on imitation learning. *Foundations and Trends® in Robotics*, 7(1-2):1–179, 2018.

F. Otto, P. Becker, V. A. Ngo, H. C. M. Ziesche, and G. Neumann. Differentiable trust region layers for deep reinforcement learning. In *International Conference on Learning Representations*, 2021.

F. Otto, O. Celik, H. Zhou, H. Ziesche, V. A. Ngo, and G. Neumann. Deep black-box reinforcement learning with movement primitives. In *Conference on Robot Learning*, pages 1244–1265. PMLR, 2023.

F. Otto, P. Becker, V. A. Ngo, and G. Neumann. Efficient off-policy learning for high-dimensional action spaces. In *The Thirteenth International Conference on Learning Representations*, 2025.

L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

D. S. Pandey, S. Pyakurel, and Q. Yu. Be confident in what you know: Bayesian parameter efficient fine-tuning of vision foundation models. In *Advances in Neural Information Processing Systems*, volume 37, pages 44814–44844. Curran Associates, Inc., 2024.

A. Paraschos, C. Daniel, J. R. Peters, and G. Neumann. Probabilistic movement primitives. *Advances in neural information processing systems*, 26, 2013.

A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems; Network of Plausible Inference.* Morgan Kaufmann, 1988, 1988.

A. Pentland and A. Liu. Modeling and prediction of human behavior. *Neural Comput.*, 11(1):229–242, Jan. 1999. ISSN 0899-7667. doi: 10.1162/089976699300016890.

J. Peters, K. Mülling, and Y. Altun. Relative entropy policy search. In *AAAI*, pages 1607–1612. Atlanta, 2010.

D. A. Pomerleau. Efficient training of artificial neural networks for autonomous navigation. *Neural computation*, 3(1):88–97, 1991.

B. Poole, A. A. Alemi, J. Sohl-Dickstein, and A. Angelova. Improved generator objectives for gans. *arXiv preprint arXiv:1612.02780*, 2016.

B. Poole, S. Ozair, A. Van Den Oord, A. Alemi, and G. Tucker. On variational bounds of mutual information. In *International Conference on Machine Learning*, pages 5171–5180. PMLR, 2019.

A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR, 2021.

A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *Robotics: Science and Systems XIV*, 2018.

K. Rakelly, A. Zhou, C. Finn, S. Levine, and D. Quillen. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *International conference on machine learning*, pages 5331–5340. PMLR, 2019.

A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.

R. Ranganath, D. Tran, and D. Blei. Hierarchical variational models. In *International Conference on Machine Learning*, pages 324–333, 2016.

S. S. Rangapuram, M. W. Seeger, J. Gasthaus, L. Stella, Y. Wang, and T. Januschowski. Deep state space models for time series forecasting. *Advances in neural information processing systems*, 2018.

H. E. Rauch, F. Tung, and C. T. Striebel. Maximum likelihood estimates of linear dynamic systems. *AIAA journal*, 3(8):1445–1450, 1965.

D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International conference on machine learning*, pages 1278–1286. PMLR, 2014.

A. Robicquet, A. Sadeghian, A. Alahi, and S. Savarese. Learning social etiquette: Human trajectory understanding in crowded scenes, 10 2016.

R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

S. Ross, G. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.

R. Y. Rubinstein. Optimization of computer simulation models with rare events. *European Journal of Operational Research*, 99(1):89–112, 1997.

M. R. Samsami, A. Zholus, J. Rajendran, and S. Chandar. Mastering memory tasks with world models. In *The Twelfth International Conference on Learning Representations*, 2024.

S. Särkkä and Á. F. García-Fernández. Temporal parallelization of bayesian smoothers. *IEEE Transactions on Automatic Control*, 66(1):299–306, 2020.

F. Schmidt and T. Hofmann. Deep state space models for unconditional word generation. *Advances in Neural Information Processing Systems 31*, 31:6158–6168, 2018.

J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, pages 1889–1897, 2015a.

J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015b.

J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

R. Sekar, O. Rybkin, K. Daniilidis, P. Abbeel, D. Hafner, and D. Pathak. Planning to explore via self-supervised world models. In *International Conference on Machine Learning*, pages 8583–8592. PMLR, 2020.

F. Seligmann, P. Becker, M. Volpp, and G. Neumann. Beyond deep ensembles: A large-scale evaluation of bayesian deep learning under distribution shift. In *Advances in Neural Information Processing Systems*, volume 36, pages 29372–29405. Curran Associates, Inc., 2023.

S. Sengupta, M. Harris, Y. Zhang, and J. D. Owens. Scan primitives for gpu computing, 2007.

Y. Seo, D. Hafner, H. Liu, F. Liu, S. James, K. Lee, and P. Abbeel. Masked world models for visual control. In *6th Annual Conference on Robot Learning*, 2022.

Y. Seo, J. Kim, S. James, K. Lee, J. Shin, and P. Abbeel. Multi-view masked world models for visual robotic manipulation. *PMLR*, 2023.

V. Shaj, P. Becker, D. Buchler, H. Pandya, N. van Duijkeren, C. J. Taylor, M. Hanheide, and G. Neumann. Action-conditional recurrent kalman networks for forward and inverse dynamics learning. *Conference on Robot Learning*, 2020.

V. Shaj, D. Büchler, R. Sonker, P. Becker, and G. Neumann. Hidden parameter recurrent state space models for changing dynamics scenarios. In *International Conference on Learning Representations*, 2022.

M. Shridhar, L. Manuelli, and D. Fox. Cliport: What and where pathways for robotic manipulation. In *Conference on Robot Learning*, pages 894–906. PMLR, 2022.

M. Shridhar, L. Manuelli, and D. Fox. Perceiver-actor: A multi-task transformer for robotic manipulation. In *Conference on Robot Learning*, pages 785–799. PMLR, 2023.

R. Shu, T. Nguyen, Y. Chow, T. Pham, K. Than, M. Ghavamzadeh, S. Ermon, and H. Bui. Predictive coding for locally-linear control. In *International Conference on Machine Learning*. PMLR, 2020.

R. H. Shumway and D. S. Stoffer. An approach to time series smoothing and forecasting using the em algorithm. *Journal of time series analysis*, 3(4):253–264, 1982.

D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

J. T. Smith, A. Warrington, and S. Linderman. Simplified state space layers for sequence modeling. In *The Eleventh International Conference on Learning Representations*, 2022.

K. Sohn, H. Lee, and X. Yan. Learning structured output representation using deep conditional generative models. *Advances in neural information processing systems*, 28:3483–3491, 2015.

E. J. Sondik. *The optimal control of partially observable Markov processes.* Stanford University, 1971.

Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

N. Srivastava, W. Talbott, M. B. Lopez, S. Zhai, and J. M. Susskind. Robust robotic control from pixels using contrastive recurrent state-space models. In *Deep RL Workshop NeurIPS 2021*, 2021.

A. Stooke, K. Lee, P. Abbeel, and M. Laskin. Decoupling representation learning from reinforcement learning. In *International Conference on Machine Learning*, pages 9870–9879. PMLR, 2021.

J. Straub, T. Whelan, L. Ma, Y. Chen, E. Wijmans, S. Green, J. J. Engel, R. Mur-Artal, C. Ren, S. Verma, A. Clarkson, M. Yan, B. Budge, Y. Yan, X. Pan, J. Yon, Y. Zou, K. Leon, N. Carter, J. Briales, T. Gillingham, E. Mueggler, L. Pesqueira, M. Savva, D. Batra, H. M. Strasdat, R. D. Nardi, M. Goesele, S. Lovegrove, and R. Newcombe. The Replica dataset: A digital replica of indoor spaces. *arXiv preprint arXiv:1906.05797*, 2019.

M. Sugiyama, T. Suzuki, and T. Kanamori. Density-ratio matching under the bregman divergence: a unified framework of density-ratio estimation. *Annals of the Institute of Statistical Mathematics*, 64(5):1009–1044, 2012.

R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018.

E. Talvitie. Model regularization for stable sample rollouts. In *UAI*, pages 780–789, 2014.

Y. Tassa, Y. Doron, A. Muldal, T. Erez, Y. Li, D. d. L. Casas, D. Budden, A. Abdolmaleki, J. Merel, A. Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

Y. Tassa, S. Tunyasuvunakool, A. Muldal, Y. Doron, S. Liu, S. Bohez, J. Merel, T. Erez, T. Lillicrap, and N. Heess. dm_control: Software and tasks for continuous control, 2020.

A. Tirinzoni, A. Touati, J. Farebrother, M. Guzek, A. Kanervisto, Y. Xu, A. Lazaric, and M. Pirotta. Zero-shot whole-body humanoid control via behavioral foundation models. In *The Thirteenth International Conference on Learning Representations*, 2025.

N. Tishby, F. C. Pereira, and W. Bialek. The information bottleneck method. In *Proc. of the 37-th Annual Allerton Conference on Communication, Control and Computing*, 1999.

M. Tomar, U. A. Mishra, A. Zhang, and M. E. Taylor. Learning representations for pixel-based control: What matters and why? *Transactions on Machine Learning Research*, 2023.

M. Uehara, I. Sato, M. Suzuki, K. Nakayama, and Y. Matsuo. Generative adversarial nets from a density ratio estimation perspective. *arXiv preprint arXiv:1610.02920*, 2016.

U.S. Dep. of Transportation. `https://data.transportation.gov/Automobiles/Next-Generation-Simulation-NGSIM-Vehicle-Trajector/8ect-6jqj`, 2016.

A. Van Den Oord, O. Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

C. Villani et al. *Optimal transport: old and new*, volume 338. Springer, 2008.

M. Volpp, F. Flürenbrock, L. Grossberger, C. Daniel, and G. Neumann. Bayesian context aggregation for neural processes. In *International Conference on Learning Representations*, 2021.

N. Wahlström, T. B. Schön, and M. P. Deisenroth. From pixels to torques: Policy learning with deep dynamical models. *arXiv preprint arXiv:1502.02251*, 2015.

S. Wang, R. Clark, H. Wen, and N. Trigoni. Deepvo: Towards end-to-end visual odometry with deep recurrent convolutional neural networks. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 2043–2050. IEEE, 2017.

T. Wang, S. Du, A. Torralba, P. Isola, A. Zhang, and Y. Tian. Denoised mdps: Learning world models better than the world itself. In *International Conference on Machine Learning*. PMLR, 2022.

M. Watter, J. Springenberg, J. Boedecker, and M. Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. In *Advances in neural information processing systems*, pages 2746–2754, 2015.

P. J. Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

P. Wu, A. Escontrela, D. Hafner, P. Abbeel, and K. Goldberg. Daydreamer: World models for physical robot learning. In *6th Annual Conference on Robot Learning*, 2022.

T. Xiao, I. Radosavovic, T. Darrell, and J. Malik. Masked visual pre-training for motor control. *arXiv preprint arXiv:2203.06173*, 2022.

D. Yarats, R. Fergus, A. Lazaric, and L. Pinto. Reinforcement learning with prototypical representations. In *International Conference on Machine Learning*, pages 11920–11931. PMLR, 2021a.

D. Yarats, A. Zhang, I. Kostrikov, B. Amos, J. Pineau, and R. Fergus. Improving sample efficiency in model-free reinforcement learning from images. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 10674–10681, 2021b.

D. Yarats, R. Fergus, A. Lazaric, and L. Pinto. Mastering visual continuous control: Improved data-augmented reinforcement learning. In *International Conference on Learning Representations*, 2022.

L. Yingzhen and S. Mandt. Disentangled sequential autoencoder. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 5670–5679. PMLR, 10–15 Jul 2018.

B. You, O. Arenz, Y. Chen, and J. Peters. Integrating contrastive learning with dynamic models for reinforcement learning from images. *Neurocomputing*, 476:102–114, 2022. ISSN 0925-2312.

A. Zhang, R. T. McAllister, R. Calandra, Y. Gal, and S. Levine. Learning invariant representations for reinforcement learning without reconstruction. In *International Conference on Learning Representations*, 2020.

X. Zheng, M. Zaheer, A. Ahmed, Y. Wang, E. P. Xing, and A. J. Smola. State space lstm models with particle mcmc inference. *arXiv preprint arXiv:1711.11179*, 2017.

L. Zhou, M. Poli, W. Xu, S. Massaroli, and S. Ermon. Deep latent state space models for time-series generation. In *International Conference on Machine Learning*, pages 42625–42643. PMLR, 2023.

T. Zhou, M. Brown, N. Snavely, and D. G. Lowe. Unsupervised learning of depth and ego-motion from video. In *CVPR*, page 7, 2017.

C. Zhu, M. Simchowitz, S. Gadipudi, and A. Gupta. Repo: Resilient model-based reinforcement learning by regularizing posterior predictability. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

B. D. Ziebart, A. L. Maas, J. A. Bagnell, A. K. Dey, et al. Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pages 1433–1438. Chicago, IL, USA, 2008.

L. Zintgraf, K. Shiarlis, M. Igl, S. Schulze, Y. Gal, K. Hofmann, and S. Whiteson. Varibad: A very good method for bayes-adaptive deep rl via meta-learning. In *International Conference on Learning Representations*, 2020.

List of Publications

Here I present an exhaustive list of all publications I have contributed to, thus far.

Journal Publications

On Uncertainty in Deep State Space Models for Model-Based Reinforcement Learning
Philipp Becker, Gerhard Neumann
Transactions in Machine Learning Research (TMLR), 2022

Conference Publications

EDiT: Efficient Diffusion Transformers with Linear Compressed Attention
Philipp Becker, Abhinav Mehrotra, Ruchika Chavhan, Malcolm Chadwick, Luca Morreale, Mehdi Noroozi, Alberto Gil Ramos, Sourav Bhattacharya
International Conference on Computer Vision 2025 (ICCV 2025), Written during my Internship with Samsung Research UK at the AI Center in Cambridge

Geometry-aware RL for Manipulation of Varying Shapes and Deformable Objects
Tai Hoang, Huy Le, *Philipp Becker*, Ngo Anh Vien, Gerhard Neumann
13th International Conference on Learning Representations (ICLR 2025)

Efficient Off-Policy Learning for High-Dimensional Action Spaces
Fabian Otto, *Philipp Becker*, Ngo Anh Vien, Gerhard Neumann
13th International Conference on Learning Representations (ICLR 2025)

Combining Reconstruction and Contrastive Methods for Multimodal Representations in RL
Philipp Becker, Sebastian Mossburger, Fabian Otto, Gerhard Neumann
First Reinforcement Learning Conference (RLC 2024)

PointPatchRL - Masked Reconstruction Improves Reinforcement Learning on Point Clouds
Balázs Gyenes, Nikolai Franke, *Philipp Becker*, Gerhard Neumann
Eight Conference on Robot Learning (CoRL 2024)

MuTT: A Multimodal Trajectory Transformer for Robot Skills
Claudius Kienle, Benjamin Alt, Onur Celik, *Philipp Becker*, Darko Katic, Rainer Jäkel, Gerhard Neumann
The 2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2024)

Beyond Deep Ensembles: A Large-Scale Evaluation of Bayesian Deep Learning under Distribution Shift
Florian Seligmann, *Philipp Becker*, Michael Volpp, Gerhard Neumann
37th Conference on Neural Information Processing Systems (NeurIPS 2023)

Curriculum-Based Imitation of Versatile Skills
Maximilian Xiling Li, Onur Celik, *Philipp Becker*, Denis Blessing, Rudolf Lioutikov, Gerhard Neumann
2023 IEEE International Conference on Robotics and Automation (ICRA 2023)

Accurate Bayesian Meta-Learning by Accurate Task Posterior Inference
Michael Volpp, Philipp Dahlinger, *Philipp Becker*, Christian Daniel, Gerhard Neumann
11th International Conference on Learning Representations (ICLR 2023)

Inferring Versatile Behavior from Demonstrations by Matching Geometric Descriptors
Niklas Freymuth, Nicolas Schreiber, *Philipp Becker*, Aleksandar Taranovic, Gerhard Neumann
6th Conference on Robot Learning (CoRL 2022)

End-to-End Learning of Hybrid Inverse Dynamics Models for Precise and Compliant Impedance Control
Moritz Reuss, Niels van Duijkeren, Robert Krug, *Philipp Becker*, Vaisakh Shaj, Gerhard Neumann
Robotics: Science and Systems XVIII, (RSS 2022)

Hidden Parameter Recurrent State Space Models For Changing Dynamics Scenarios
Vaisakh Shaj, Dieter Buchler, Rohit Sonker, *Philipp Becker*, Gerhard Neumann
10th International Conference on Learning Representations (ICLR 2022)

Specializing Versatile Skill Libraries using Local Mixture of Experts
Onur Celik, Dongzhuoran Zhou, Ge Li, *Philipp Becker*, Gerhard Neumann
5th Conference on Robot Learning (CoRL 2021)

Differentiable Trust Region Layers for Deep Reinforcement Learning
Fabian Otto, *Philipp Becker*, Ngo Anh Vien, Hanna Carolin Ziesche, Gerhard Neumann
9th International Conference on Learning Representations (ICLR 2021)

Action-Conditional Recurrent Kalman Networks For Forward and Inverse Dynamics Learning
Vaisakh Shaj, *Philipp Becker*, Dieter Büchler, Harit Pandya, Niels van Duijkeren, C. James Taylor, Marc Hanheide, Gerhard Neumann
4th Conference on Robot Learning (CoRL 2020)

Expected Information Maximization: Using the I-Projection for Mixture Density Estimation
Philipp Becker, Oleg Arenz, Gerhard Neumann
8th International Conference on Learning Representations (ICLR 2020)

Recurrent Kalman Networks: Factorized Inference in High-Dimensional Deep Feature Spaces
Philipp Becker, Harit Pandya, Gregor Gebhardt, Cheng Zhao, C. James Taylor, Gerhard Neumann
36th International Conference on Machine Learning, (ICML 2019), Written and published before starting my PhD studies

Workshop and other Non-Archival Publications

AMBER: Adaptive Mesh Generation by Iterative Mesh Resolution Prediction
Niklas Freymuth, Tobias Würth, Nicolas Schreiber, Balazs Gyenes, Andreas Boltres, Johannes Mitsch, Aleksandar Taranovic, Tai Hoang, Philipp Dahlinger, *Philipp Becker*, Luise Kärger, Gerhard Neumann *ArXiv, 2025*

Adaptive World Models: Learning Behaviors by Latent Imagination Under Non-Stationarity
Emiliyan Gospodinov, Vaisakh Shaj, *Philipp Becker*, Stefan Geyer, Gerhard Neumann
Workshop on Adaptive Foundation Models @ NeurIPS 2024

KalMamba: Towards Efficient Probabilistic State Space Models for RL under Uncertainty
Philipp Becker, Niklas Freymuth, Gerhard Neumann
Aligning Reinforcement Learning Experimentalists and Theorists Workshop @ ICML, 2024
Next Generation of Sequence Modeling Architectures Workshop @ ICML, 2024
(Workshop on) Training Agents with Foundation Models @ RLC, 2024

Iterative Sizing Field Prediction for Adaptive Mesh Generation From Expert Demonstrations
Niklas Freymuth, Philipp Dahlinger, Tobias Würth, *Philipp Becker*, Aleksandar Taranovic, Onno Grönheim, Luise Kärger, Gerhard Neumann
AI for Science Workshop @ ICML, 2024

Information-Theoretic Trust Regions for Stochastic Gradient-Based Optimization
Philipp Dahlinger, *Philipp Becker*, Maximilian Hüttenrauch, Gerhard Neumann
Workshop on Optimization for Machine Learning @ NeurIPS, 2023

Switching Recurrent Kalman Networks
Giao Nguyen-Quynh, *Philipp Becker*, Chen Qiu, Maja Rudolph, Gerhard Neumann
Machine Learning for Autonomous Driving Workshop @ NeurIPS 2021

Versatile Inverse Reinforcement Learning via Cumulative Rewards
Niklas Freymuth, *Philipp Becker*, Gerhard Neumann
Workshop on Robot Learning: Self-Supervised and Lifelong Learning @ NeurIPS, 2021

List of Supervised Student Theses

During my time at the Karlsruhe Institute of Technology, I had the privilege to supervise the following student theses. For some of these, I co-supervised with fellow PhD students as well as industrial partners.

Master Theses

Erwin Müller, Model-based Reinforcement Learning of Diverse Skills, *Co-Supervised with Onur Celik*

Keven Sahin, Offline Reinforcement Learning for Hydraulic Excavator Control, *in cooperation with Robert Bosch GmbH, Co-Supervised with Vaisakh Shaj*

Sven Hillenkötter, Model-Based Reinforcement Learning for Challenging Manipulation Tasks

Yasin Tatar, Investigating the Amortization Gap in Variational Inference for State Space Models

Claudius Kienle, Multimodal Transformer for Zero-Shot Optimization of Robot Manipulation Tasks Across Environments, *in cooperation with ArtiMinds Robotics GmbH, Co-Supervised with Onur Celik*

Ruben Jacob, Episodic Reinforcement Learning with Transformer Models, *Co-Supervised with Vaisakh Shaj*

Aina Galofre, Improving Uncertainty Estimation for Model-Based Reinforcement Learning

Christian Schorr, Improving the Soft Actor-Critic with Information Theoretic Trust Regions, *Co-Supervised with Onur Celik*

Sebastian Mossburger, Self-Supervised Model-Based Reinforcement Learning under Partial Observability, *Name on thesis: Sebastian Markgraf*

Maximilian Xiling Li, Imitation Learning for Versatile Robot Manipulation, *Co-Supervised with Onur Celik*

Ahmed Agha, Improved Trust Regions for Adversarial Imitation Learning, *in cooperation with Technische Universität München, Co-Supervised with Hanna Ziesche and Fabian Otto*

Moritz Reuss, Hybrid Inverse Dynamics Models for Robot Impedance Control, *in cooperation with Robert Bosch GmbH, Co-Supervised with Vaisakh Shaj*

Giao Nguyen-Quynh, Kalman Mixture Networks, *in cooperation with Robert Bosch GmbH*

Philipp Dahlinger, Information Theoretic Trust Regions for Gradient Descent, *Follow up of one year research project, Co-Supervised with Maximilian Hüttenrauch*

Niklas Freymuth, Inverse Reinforcement Learning for Highly-Versatile Behavior

Carmen Eitel, Deep Reinforcement Learning under Partial Observability using Kalman Filtering

Bachelor Theses

Felix Ning, Offline Model-Based Reinforcement Learning from Visual Observations

Ralf Herp, Diffusion-based Imitation Learning for Contact Rich Manipulation, *in cooperation with FZI Research Center for Information Technology*

Florian Seligmann, Benchmarking Modern Bayesian Deep Learning Algorithms, *Followed by one year research project, Co-Supervised with Michael Volpp*

Roman Freiberg, Actor-Critic Methods for Robotics, *Co-Supervised with Onur Celik*

Stefan Krieg, Tackling Offline Reinforcement Learning by Model Learning

A. Appendix for Chapter 3

A.1. Derivations

Derivations of the upper bound stated in Equation 3.1. We assume latent variable models $q(x) = \int q(x|z)q(z)dz$ and use the identities $q(x|z)q(z) = q(z|x)q(x)$ and $\log q(x) = \log q(x|z)q(z) - \log q(z|x)$

$$
\begin{aligned}
&\mathrm{KL}\left(q(x)||p(x)\right) = \int q(x)\log\frac{q(x)}{p(x)}dx = \iint q(x|z)q(z)\log\frac{q(x)}{p(x)}dzdx \\
&= \iint q(x|z)q(z)\left(\log\frac{q(x|z)q(z)}{p(x)} - \log q(z|x)\right)dzdx \\
&= \iint q(x|z)q(z)\left(\log\frac{q(x|z)q(z)}{p(x)} - \log q(z|x) + \log\tilde{q}(z|x) - \log\tilde{q}(z|x)\right)dzdx \\
&= \iint q(x|z)q(z)\left(\log\frac{q(x|z)q(z)}{p(x)} - \log\tilde{q}(z|x)\right)dzdx \\
&\quad - \iint q(x|z)q(z)\left(\log q(z|x) - \log\tilde{q}(z|x)\right)dzdx \\
&= \iint q(x|z)q(z)\left(\log\frac{q(x|z)q(z)}{p(x)} - \log\tilde{q}(z|x)\right)dzdx - \int q(x)\int q(z|x)\log\frac{q(z|x)}{\tilde{q}(z|x)}dzdx \\
&= U(q,\tilde{q},p) - \mathbb{E}_{q(x)}\left[\mathrm{KL}\left(q(z|x)||\tilde{q}(z|x)\right)\right].
\end{aligned}
$$

After plugging the E-Step, i.e., $\tilde{q}(z|x) = q_t(x|z)q_t(z)/q_t(x)$, into the objective it simplifies to

$$
\begin{aligned}
&U(q,\tilde{q},p) \\
&= \iint q(x|z)q(z)\left(\log\frac{q(x|z)q(z)}{p(x)} - \log\frac{q_t(x|z)q_t(z)}{q_t(x)}\right)dzdx \\
&= \iint q(x|z)q(z)\left(\log q(x|z) + \log q(z) - \log p(x) - \log q_t(x|z) - \log q_t(z) + \log q_t(x)\right)dzdx \\
&= \iint q(x|z)q(z)\left(\log\frac{q_t(x)}{p(x)} + \log\frac{q(x|z)}{q_t(x|z)} + \log\frac{q(z)}{q_t(z)}\right)dzdx \\
&= \int q(z)\left(\int q(x|z)\left(\log\frac{q_t(x)}{p(x)} + \log\frac{q(x|z)}{q_t(x|z)}\right)dx + \log\frac{q(z)}{q_t(z)}\right)dz \\
&= \int q(z)\int q(x|z)\log\frac{q_t(x)}{p(x)}dxdz + \int q(z)\int q(x|z)\log\frac{q(x|z)}{q_t(x|z)}dxdz + \int q(z)\log\frac{q(z)}{q_t(z)}dz \\
&= \iint q(x|z)q(z)\log\frac{q_t(x)}{p(x)}dzdx + \mathbb{E}_{q(z)}\left[\mathrm{KL}\left(q(x|z)||q_t(x|z)\right)\right] + \mathrm{KL}\left(q(z)||q_t(z)\right),
\end{aligned}
$$

which concludes the derivation of upper bound of latent variable models.

A.1.1. Derivations Conditional Upper Bound

By introducing an auxiliary distribution $\tilde{q}(z|x,y)$ the upper bound to the expected KL for conditional latent variable models $q(x|y) = \int q(x|z,y)q(z|y)dz$ can be derived by

$$
\begin{aligned}
&\mathbb{E}_{p(y)}\left[\mathrm{KL}\left(q(x|y)||p(x|y)\right)\right] = \iint p(y)q(x|y)\log\frac{q(x|y)}{p(x|y)}dxdy \\
=&\int p(y)\iint q(x|z,y)q(z|y)\left(\log\frac{q(x|z,y)q(z|y)}{p(x|y)} - \log q(z|x,y)\right)dzdxdy \\
=&\int p(y)\iint q(x|z,y)q(z|y) \\
&\cdot\left(\log\frac{q(x|z,y)q(z|y)}{p(x|y)} - \log q(z|x,y) + \log\tilde{q}(z|x,y) - \log\tilde{q}(z|x,y)\right)dzdxdy \\
=&\int p(y)\iint q(x|z,y)q(z|y)\left(\log\frac{q(x|z,y)q(z|y)}{p(x|y)} - \log\tilde{q}(z|x,y)\right)dzdxdy \\
&-\int p(y)\iint q(x|z,y)q(z,y)\left(\log q(z|x,y) - \log\tilde{q}(z|x,y)\right)dzdxdy \\
=&\int p(y)\iint q(x|z,y)q(z|y)\left(\log\frac{q(x|z,y)q(z|y)}{p(x|y)} - \log\tilde{q}(z|x,y)\right)dzdxdy \\
&-\iint p(y)q(x|y)\int q(z|x,y)\log\frac{q(z|x,y)}{\tilde{q}(z|x,y)}dzdxdy \\
=&U(q,\tilde{q},p) - \mathbb{E}_{p(y),q(x|y)}\left[\mathrm{KL}\left(q(z|x,y)||\tilde{q}(z|x,y)\right)\right].
\end{aligned}
$$

During the E-step the bound is tightened by setting $\tilde{q}(z|x,y) = q_t(x|z,y)q_t(z|y)/q_t(x|y)$.

$$
\begin{aligned}
&U(q,\tilde{q},p) \\
=&\int p(y)\iint q(x|z,y)q(z|y)\left(\log\frac{q(x|z,y)q(z|y)}{p(x|y)} - \log\frac{q_t(x|z,y)q_t(z|y)}{q_t(x|y)}\right)dzdxdy \\
=&\int p(y)\iint q(x|z,y)q(z|y)(\log q(x|z,y) + \log q(z|y) - \log p(x|y) - \log q_t(x|z,y) \\
&-\log q_t(z|y) + \log q_t(x|y))dzdxdy \\
=&\int p(y)\iint q(x|z,y)q(z|y)\left(\log\frac{q_t(x|y)}{p(x|y)} + \log\frac{q(x|z,y)}{q_t(x|z,y)} + \log\frac{q(z|y)}{q_t(z|y)}\right)dzdxdy \\
=&\int p(y)\int q(z|y)\left(\int q(x|z,y)\left(\log\frac{q_t(x|y)}{p(x|y)} + \log\frac{q(x|z,y)}{q_t(x|z,y)}\right)dx + \log\frac{q(z|y)}{q_t(z|y)}\right)dzdy \\
=&\int p(y)\int q(z|y)\int q(x|z,y)\log\frac{q_t(x|y)}{p(x|y)}dxdzdy \\
&+\int p(y)\int q(z|y)\int q(x|z,y)\log\frac{q(x|z,y)}{q_t(x|z,y)}dxdzdy + \int p(y)\int q(z|y)\log\frac{q(z|y)}{q_t(z|y)}dzdy \\
=&\iiint p(y)q(z|y)q(x|z,y)\log\frac{q_t(x|y)}{p(x|y)}dxdzdy \\
&+\mathbb{E}_{p(y),q(z|y)}\left[\mathrm{KL}\left(q(x|z,y)||q_t(x|z,y)\right)\right] + \mathbb{E}_{p(y)}\left[\mathrm{KL}\left(q(z|y)||q_t(z|y)\right)\right],
\end{aligned}
$$

which concludes the derivation of the upper bound for conditional latent variable models.

A.1.2. Using MORE for closed form updates for GMMs

The MORE algorithm, as introduced by Abdolmaleki et al. (2015), can be used to solve optimization problems of the following form

$$\arg\max_{q(x)} \mathbb{E}_{q(x)}[f(x)] \quad \text{s.t.} \quad \text{KL}\left(q(x)||q_{\text{old}}(x)\right) \leq \epsilon$$

for an exponential family distribution $q(x)$, some function $f(x)$, and an upper bound on the allowed change, ϵ. Abdolmaleki et al. (2015) show that the optimal solution is given by

$$q(x) \propto q_{\text{old}}(x) \exp\left(\frac{f(x)}{\eta}\right) = \exp\left(\frac{\eta \log q_{\text{old}}(x) + f(x)}{\eta}\right),$$

where η denotes the Lagrangian multiplier corresponding to the KL constraint. In order to obtain this Lagrangian multiplier, the following, convex, dual function has to be minimized

$$g(\eta) = \eta\epsilon + \eta \log \int \exp\left(\frac{\eta \log q_{\text{old}}(x) + f(x)}{\eta}\right) dx. \tag{A.1}$$

For discrete distributions, such as the categorical distribution used to represent the coefficients of a GMM, we can directly work with those equations. For continuous distributions, Abdolmaleki et al. (2015) propose approximating $f(x)$ with a local surrogate. The features to fit this surrogate are chosen such that they are compatible (Kakade, 2002), i.e., of the same form as the distributions sufficient statistics. For multivariate Gaussians, the sufficient statistics are squared features and thus the surrogate compatible to such a Gaussian distribution is given by

$$\hat{f}(x) = -\frac{1}{2}\mathbf{x}^T\hat{\mathbf{F}}\mathbf{x} + \hat{\mathbf{f}}^T\mathbf{x} + f_0.$$

The parameters of this surrogate can now be used to update the natural parameters of the Gaussian, i.e, the precision matrix $\mathbf{Q} = \Sigma^{-1}$ and $\mathbf{q} = \Sigma^{-1}\boldsymbol{\mu}$ by

$$\mathbf{Q} = \mathbf{Q}_t + \frac{1}{\eta}\hat{\mathbf{F}} \quad \text{and} \quad \mathbf{q} = \mathbf{q}_t + \frac{1}{\eta}\hat{\mathbf{f}}.$$

In order to apply the MORE algorithm to solve the optimization problems stated in Equation 3.4 and Equation 3.5 we make two trivial modifications. First, we invert the signs in Equation 3.4 and Equation 3.5, as we are now maximizing. Second, to account for the additional KL term in our objectives, we add 1 to η, everywhere except the first term of the sum in Equation A.1.

This is very similar to the approach Arenz et al. (2018) use in their VIPS algorithm. The main difference lies in how $f(x)$ is defined, where for EIM it is the log density ratio $f(x) = \frac{q_t(x)}{p_t(x)}$, in its original formulation, VIPS uses $f(x) = \log p(x) + \log q_t(z|x)$. Instead of having to account for an additional KL term, this formulation requires us to account for an additional entropy term in the dual function. Thus, we need to add a 1 to η, everywhere except the first term of the sum in Equation A.1 and the $\eta \log q_t(x)$ term nominator in Equation A.1

A.2. Hyperparameters

In all experiments, we realize the density ratio estimator as fully connected neural networks which we train using Adam (Kingma and Ba, 2014) and early stopping using a validation set.

Comparison to Generative Adversarial Approaches and Ablation Study

- Data: $10,000$ Train Samples, $5,000$ Test Samples, $5,000$ Validation samples (for early stopping the density ratio estimator)
- Density Ratio Estimator (EIM) / Variational function $V(x)$ (f-GAN): 3 fully connected layers, 50 neurons each, trained with L2 regularization with factor 0.001, early stopping and batch size $1,000$
- Updates EIM: MORE-like updates with $\epsilon = 0.05$ for components and coefficients, $1,000$ samples per component and update
- Updates FGAN: Iterate single update steps for generator and discriminator using learning rates of $1e-3$ and batch size of $1,000$.

Line Reaching with Planar Robot

- Data: $10,000$ train samples, $5,000$ test samples, $5,000$ validation samples (for early stopping the density ratio estimator)
- Density Ratio Estimation: 2 fully connected layers of width 100, early stopping and batch size $1,000$
- Updates: MORE-like updates with $\epsilon = 0.005$ for components and coefficients, $1,000$ samples per component and update

Pedestrian and Traffic Prediction

- Data SDD: $7,500$ train samples, $3,801$ test samples, $3,801$ validation samples (for early stopping the density ratio estimator)
- Data NGS: $10,000$ train samples, $5,000$ test samples, $5,000$ validation samples (for early stopping the density ratio estimator)
- Density Ratio Estimation: 3 fully connected layers of width 256, trained with L2 regularization with factor 0.0005 early stopping and batch size $1,000$.
- Updates: MORE-like updates with $\epsilon = 0.01$ for components and coefficients, $1,000$ samples per component and update

Obstacle Avoidance

- Data: $1,000$ train contexts with 10 samples each, 500 test contexts with 10 samples each, 500 validation contexts with 10 samples each (for early stopping the density ratio estimator).
- Density Ratio Estimation: 3 fully connected layers of width 256, trained with L2 regularization with factor 0.0005 early stopping and batch size $1,000$.
- Component and Gating networks: 2 fully connected layer of width 64 for each component and the gating. Trained with Adam ($\alpha = 1e-3, \beta_0 = 0.5$) for 10 epochs in each iteration.

A.3. Further Discussion on Prior Works

A.3.1. Equality of f-GAN and b-GAN

Both the f-GAN (Nowozin et al., 2016) and the b-GAN (Uehara et al., 2016) yield the same objective for the I-Projection.

We start with the f-GAN. Nowozin et al. (2016) propose the following adversarial objective, based on a variatonal bound for f-divergences (Nguyen et al., 2010)

$$\arg\min_{q(x)} \arg\max_{V(x)} F(q(x), V(x)) = \mathbb{E}_{p(x)}\left[g(V(x))\right] - \mathbb{E}_{q(x)}\left[f^*(g(V(x)))\right].$$

Here $V(x)$ denotes a neural network with linear output, $g(v)$ the output activation and $f^*(t)$ the Fenchel conjugate (Hiriart-Urruty and Lemaréchal, 2012) of $f(u)$, i.e., the generator function of the f-divergence. For the I-Projection $f(u) = -\log u$ and $f^*(t) = -1 - \log(t)$. In theory, the only restriction posed on the choice of $g(v)$ is that it outputs only values within the domain of $f^*(t)$ Nowozin et al. (2016) suggest g's for various $f-$ divergences and chose exclusively monotony increasing functions which output large values for samples that are believed to be from the data distribution. For the I-Projection they suggest $g(v) = -exp(-v)$. Thus the f-GAN objective for the I-Projection is given by

$$\arg\min_{q(x)} \arg\max_{V(x)} F(q(x), V(x)) = -\mathbb{E}_{p(x)}\left[\exp(-V(x)\right] + \mathbb{E}_{q(x)}\left[1 - V(x)\right].$$

The b-GAN objective follows from the density ratio estimation framework given by Sugiyama et al. (2012) and is given by

$$\arg\min_{q(x)} \arg\max_{r(x)} \mathbb{E}_{p(x)}\left[f'(r(x))\right] - \mathbb{E}_{q(x)}\left[f'(r(x))r(x) - f(r(x))\right]$$

Here $f'(u)$ denotes the derivative of $f(u)$ and $r(x)$ denotes an density ratio estimator. We need to enforce $r(x) > 0$ for all x to obtain a valid density ratio estimate. In practice this is usually done by learning $r_l(x) = \log r(x)$ instead. Plugging $r_l(x)$, $f(u)$ and $f'(u) = 1/u$ into the general b-GAN objective yields

$$\arg\min_{q(x)} \arg\max_{r_l(x)} F(q(x), r_l(x)) = -\mathbb{E}_{p(x)}\left[\exp(-r_l(x))\right] + \mathbb{E}_{q(x)}\left[1 - r_l(x)\right].$$

Which is the same objective as the f-GAN uses. Yet, f-GAN and b-GAN objectives are not identical for arbitrary f-divergences.

B. Appendix for Chapter 4

B.1. Simplified Kalman Filter Formulas

As stated above the simple latent observation model $\mathbf{H} = \begin{bmatrix} \mathbf{I}_m & \mathbf{0}_{m\times(n-m)} \end{bmatrix}$, as well as the assumed factorization of the covariance matrices allow us to simplify the Kalman Filter equations.

B.1.1. Notation

In the following derivations we neglect the time indices t and $t+1$ for brevity. For any matrix $\mathbf{M}$, $\hat{\mathbf{M}}$ denotes a diagonal matrix with the same diagonal as $\mathbf{M}$, $\mathbf{m}$ denotes a vector containing those diagonal elements and $M^{(ij)}$ denotes the entry at row i and column j. Similarly, $v^{(i)}$ denotes the i-th entry of a vector $\mathbf{v}$. The point wise product between two vectors of same length (Hadamat Product) will be denoted by $\odot$ and the point wise division by $\oslash$.

B.1.2. Prediction Step

$$\text{Mean:} \qquad \mathbf{z}^{-} = \mathbf{A}\mathbf{z}^{+}$$
$$\text{Covariance:} \qquad \Sigma^{-} = \mathbf{A}\Sigma^{+}\mathbf{A}^{T} + \hat{\Sigma}^{\text{dyn}}$$

The computation of the mean can not be further simplified, however, depending on the state size and bandwidth, sparse matrix multiplications may be exploited. For the covariance, let $\mathbf{T} = \mathbf{A}\Sigma^{+}$. Then,

$$\mathbf{T} = \mathbf{A}\Sigma^{+} = \begin{bmatrix} \mathbf{B}_{11} & \mathbf{B}_{12} \\ \mathbf{B}_{21} & \mathbf{B}_{22} \end{bmatrix} \begin{bmatrix} \hat{\Sigma}^{\text{u},+} & \hat{\Sigma}^{\text{s},+} \\ \hat{\Sigma}^{\text{s},+} & \hat{\Sigma}^{\text{l},+} \end{bmatrix} = \begin{bmatrix} \mathbf{B}_{11}\hat{\Sigma}^{\text{u},+} + \mathbf{B}_{12}\hat{\Sigma}^{\text{s},+} & \mathbf{B}_{11}\hat{\Sigma}^{\text{s},+} + \mathbf{B}_{12}\hat{\Sigma}^{\text{l},+} \\ \mathbf{B}_{21}\hat{\Sigma}^{\text{u},+} + \mathbf{B}_{22}\hat{\Sigma}^{\text{s},+} & \mathbf{B}_{21}\hat{\Sigma}^{\text{s},+} + \mathbf{B}_{22}\hat{\Sigma}^{\text{l},+} \end{bmatrix}$$

and

$$\begin{aligned} \Sigma^{-} = \mathbf{T}\mathbf{A}^{T} + \hat{\Sigma}^{\text{dyn}} &= \begin{bmatrix} \mathbf{B}_{11}\hat{\Sigma}^{\text{u},+} + \mathbf{B}_{12}\hat{\Sigma}^{\text{s},+} & \mathbf{B}_{11}\hat{\Sigma}^{\text{s},+} + \mathbf{B}_{12}\hat{\Sigma}^{\text{l},+} \\ \mathbf{B}_{21}\hat{\Sigma}^{\text{u},+} + \mathbf{B}_{22}\hat{\Sigma}^{\text{s},+} & \mathbf{B}_{21}\hat{\Sigma}^{\text{s},+} + \mathbf{B}_{22}\hat{\Sigma}^{\text{l},+} \end{bmatrix} \begin{bmatrix} \mathbf{B}_{11}^{T} & \mathbf{B}_{21}^{T} \\ \mathbf{B}_{12}^{T} & \mathbf{B}_{22}^{T} \end{bmatrix} + \hat{\Sigma}^{\text{dyn}} \\ &= \begin{bmatrix} \hat{\Sigma}^{\text{u},-} & \hat{\Sigma}^{\text{s},-} \\ \hat{\Sigma}^{\text{s},-} & \hat{\Sigma}^{\text{l},-} \end{bmatrix}, \end{aligned}$$

with

$$\begin{aligned}
\Sigma^{\mathrm{u},-} &= \left(\mathbf{B}_{11}\hat{\Sigma}^{\mathrm{u},+} + \mathbf{B}_{12}\hat{\Sigma}^{\mathrm{s},+}\right)\mathbf{B}_{11}^{T} + \left(\mathbf{B}_{11}\hat{\Sigma}^{\mathrm{s},+} + \mathbf{B}_{12}\hat{\Sigma}^{\mathrm{l},+}\right)\mathbf{B}_{12}^{T} + \hat{\Sigma}^{\mathrm{u,dyn}} \\
&= \mathbf{B}_{11}\hat{\Sigma}^{\mathrm{u},+}\mathbf{B}_{11}^{T} + \mathbf{B}_{12}\hat{\Sigma}^{\mathrm{s},+}\mathbf{B}_{11}^{T} + \mathbf{B}_{11}\hat{\Sigma}^{\mathrm{s},+}\mathbf{B}_{12}^{T} + \mathbf{B}_{12}\hat{\Sigma}^{\mathrm{l},+}\mathbf{B}_{12}^{T} + \hat{\Sigma}^{\mathrm{u,dyn}} \\
\Sigma^{\mathrm{l},-} &= \left(\mathbf{B}_{21}\hat{\Sigma}^{\mathrm{u},+} + \mathbf{B}_{22}\hat{\Sigma}^{\mathrm{s},+}\right)\mathbf{B}_{21}^{T} + \left(\mathbf{B}_{21}\hat{\Sigma}^{\mathrm{s},+} + \mathbf{B}_{22}\hat{\Sigma}^{\mathrm{l},+}\right)\mathbf{B}_{22}^{T} + \hat{\Sigma}^{\mathrm{l,dyn}} \\
&=\mathbf{B}_{21}\hat{\Sigma}^{\mathrm{u},+}\mathbf{B}_{21}^{T} + \mathbf{B}_{22}\hat{\Sigma}^{\mathrm{s},+}\mathbf{B}_{21}^{T} + \mathbf{B}_{21}\hat{\Sigma}^{\mathrm{s},+}\mathbf{B}_{22}^{T} + \mathbf{B}_{22}\hat{\Sigma}^{\mathrm{l},+}\mathbf{B}_{22}^{T} + \hat{\Sigma}^{\mathrm{l,dyn}} \\
\Sigma^{\mathrm{s},-} &= \left(\mathbf{B}_{21}\hat{\Sigma}^{\mathrm{u},+} + \mathbf{B}_{22}\hat{\Sigma}^{\mathrm{s},+}\right)\mathbf{B}_{11}^{T} + \left(\mathbf{B}_{21}\hat{\Sigma}^{\mathrm{s},+} + \mathbf{B}_{22}\hat{\Sigma}^{\mathrm{l},+}\right)\mathbf{B}_{12}^{T} \\
&=\mathbf{B}_{21}\hat{\Sigma}^{\mathrm{u},+}\mathbf{B}_{11}^{T} + \mathbf{B}_{22}\hat{\Sigma}^{\mathrm{s},+}\mathbf{B}_{11}^{T} + \mathbf{B}_{21}\hat{\Sigma}^{\mathrm{s},+}\mathbf{B}_{12}^{T} + \mathbf{B}_{22}\hat{\Sigma}^{\mathrm{l},+}\mathbf{B}_{12}^{T}
\end{aligned}$$

Since we are only interested in the diagonal parts of $\Sigma^{\mathrm{u},-}, \Sigma^{\mathrm{l},-}$ and $\Sigma^{\mathrm{s},-}$ i.e. $\hat{\Sigma}^{\mathrm{u},-}, \hat{\Sigma}^{\mathrm{l},-}$ and $\hat{\Sigma}^{\mathrm{s},-}$, we can further simplify these equations by realizing two properties of the terms above. First, for any matrix $\mathbf{M}, \mathbf{N}$ and a diagonal matrix $\hat{\Sigma}$ it holds that

$$\left(\mathbf{M}\hat{\Sigma}\mathbf{N}^{T}\right)^{(ii)} = \sum_{k=1}^{n} A^{(ik)}B^{(ik)}\sigma^{(k)} = \left(\mathbf{N}\hat{\Sigma}\mathbf{M}^{T}\right)_{ii}.$$

Hence, we can simplify the equations for the upper and lower part to

$$\begin{aligned}
\Sigma^{u,-} &=\mathbf{B}_{11}\hat{\Sigma}^{\mathrm{u},+}\mathbf{B}_{11}^{T} + 2\cdot\mathbf{B}_{12}\hat{\Sigma}^{\mathrm{s},+}\mathbf{B}_{11}^{T} + \mathbf{B}_{12}\hat{\Sigma}^{\mathrm{l},+}\mathbf{B}_{12}^{T} + \hat{\Sigma}^{\mathrm{u,dyn}}, \\
\Sigma^{\mathrm{l},-} &=\mathbf{B}_{21}\hat{\Sigma}^{\mathrm{u},+}\mathbf{B}_{21}^{T} + 2\cdot\mathbf{B}_{22}\hat{\Sigma}^{\mathrm{s},+}\mathbf{B}_{21}^{T} + \mathbf{B}_{22}\hat{\Sigma}^{\mathrm{l},+}\mathbf{B}_{22}^{T} + \hat{\Sigma}^{\mathrm{l,dyn}}.
\end{aligned}$$

Second, since we are only interested in the diagonal of the result it is sufficient to compute only the diagonals of the individual parts of the sums which are almost all of the same structure i.e. $\mathbf{S} = \mathbf{M}\hat{\Sigma}^{+}\mathbf{N}^{T}$. Let $\mathbf{T} = \mathbf{M}\hat{\Sigma}^{+}$, then each element of $\mathbf{T}$ can be computed as

$$T^{(ij)} = \sum_{k=1}^{n} M^{(ik)}\hat{\Sigma}^{(kj)} = M^{(ij)}\sigma^{(j)}.$$

Consequently, the elements of $\mathbf{S} = \mathbf{T}\mathbf{N}^{T}$ can be computed as

$$S^{(ij)} = \sum_{k=1}^{n} T^{(ik)}N^{(jk)} = \sum_{k=1}^{n} M^{(ik)}\sigma_{k}N^{(jk)}.$$

Ultimately, we are not interested in $\mathbf{S}$ but only in $\hat{\mathbf{S}}$

$$\hat{S}^{(ii)} = \sum_{k=1}^{n} M^{(ik)}N^{(ik)}\sigma^{(k)}.$$

Using this we obtain can obtain the entries of $\boldsymbol{\sigma}^{u,-}$, $\boldsymbol{\sigma}^{l,-}$ and $\boldsymbol{\sigma}^{s,-}$ by

$$\sigma^{u,-,(i)} = \sum_{k=1}^{m} \left(B_{11}^{(ik)}\right)^2 \sigma^{u,+,(i)} + 2\sum_{k=1}^{m} B_{11}^{(ik)} B_{12}^{(ik)} \sigma^{s,+,(i)} + \sum_{k=1}^{m} \left(B_{12}^{(ik)}\right)^2 \sigma^{l,+,(i)} + \sigma^{u,\mathrm{dyn},(i)} \tag{B.1}$$

$$\sigma^{l,-,(i)} = \sum_{k=1}^{m} \left(B_{21}^{(ik)}\right) \sigma^{u,+,(i)} + 2\sum_{k=1}^{m} B_{22}^{(ik)} B_{21}^{(ik)} \sigma^{s,+,(i)} + \sum_{k=1}^{m} \left(B_{22}^{(ik)}\right)^2 \sigma^{l,+,(i)} + \sigma^{l,\mathrm{dyn},(i)} \tag{B.2}$$

$$\sigma^{s,-,(i)} = \sum_{k=1}^{m} B_{21}^{(ik)} B_{11}^{(ik)} \sigma^{u,+,(i)} + \sum_{k=1}^{m} B_{22}^{(ik)} B_{11}^{(ik)} \sigma^{s,+,(i)} \ldots \tag{B.3}$$

$$\ldots + \sum_{k=1}^{m} B_{21}^{(ik)} B_{12}^{(ik)} \sigma^{s,+,(i)} + \sum_{k=1}^{m} B_{22}^{(ik)} B_{12}^{(ik)} \sigma^{l,+,(i)},$$

which can be implemented efficiently using elementwise matrix multiplication and sum reduction.

B.1.3. Update Step

Kalman Gain	$\mathbf{Q} = \Sigma^{-}\mathbf{H}^T\left(\mathbf{H}\Sigma^{-}\mathbf{H}^T + \Sigma^{\mathrm{obs}}\right)^{-1}$
Mean	$\boldsymbol{\mu}^{+} = \mathbf{z}^{-} + \mathbf{Q}\left(\mathbf{w} - \mathbf{H}\mathbf{z}^{-}\right)$
Covariance	$\Sigma^{+} = \left(\mathbf{I} - \mathbf{Q}\mathbf{H}\right)\Sigma^{-}$

First, note that

$$\Sigma^{-}\mathbf{H}^T = \begin{bmatrix} \hat{\Sigma}^{\mathrm{u},-} \\ \hat{\Sigma}^{\mathrm{s},-} \end{bmatrix}$$

and

$$\mathbf{H}\Sigma^{-}\mathbf{H}^T + \hat{\Sigma}^{\mathrm{obs}} = \hat{\Sigma}^{u,-} + \hat{\Sigma}^{\mathrm{obs}}$$

and thus the computation of the Kalman Gain only involves diagonal matrices. Hence the Kalman Gain matrix also consists of two diagonal matrices, i.e., $\mathbf{Q} = \begin{bmatrix} \hat{\mathbf{Q}}^{\mathrm{u}} \\ \hat{\mathbf{Q}}^{\mathrm{l}} \end{bmatrix}$ whose diagonals can be computed by

$$\mathbf{q}^{\mathrm{u}} = \boldsymbol{\sigma}^{\mathrm{u},-} \oslash \left(\boldsymbol{\sigma}^{\mathrm{u},-} + \boldsymbol{\sigma}^{\mathrm{obs}}\right) \tag{B.4}$$

$$\text{and} \quad \mathbf{q}^{\mathrm{l}} = \boldsymbol{\sigma}^{\mathrm{s},-} \oslash \left(\boldsymbol{\sigma}^{\mathrm{u},-} + \boldsymbol{\sigma}^{\mathrm{obs}}\right). \tag{B.5}$$

Using this result, the mean update can be simplified to

$$\mathbf{z}^{+} = \mathbf{z}^{-} + \begin{bmatrix} \mathbf{q}_{\mathrm{u}} \\ \mathbf{q}_{\mathrm{l}} \end{bmatrix} \odot \begin{bmatrix} \mathbf{w} - \mathbf{z}^{\mathrm{u},-} \\ \mathbf{w} - \mathbf{z}^{\mathrm{u},-} \end{bmatrix}. \tag{B.6}$$

For the covariance we get:

$$\begin{bmatrix} \hat{\Sigma}^{\mathrm{u},+} & \hat{\Sigma}^{\mathrm{s},+} \\ \hat{\Sigma}^{\mathrm{s},+} & \hat{\Sigma}^{\mathrm{l},+} \end{bmatrix} = \left(\mathbf{I}_n - \mathbf{Q}\mathbf{H}\right)\Sigma^{-} = \begin{bmatrix} \mathbf{I}_m - \hat{\mathbf{Q}}^{\mathrm{u}} & \mathbf{0}_{m\times m} \\ -\hat{\mathbf{Q}}^{\mathrm{l}} & \mathbf{I}_m \end{bmatrix} \begin{bmatrix} \hat{\Sigma}^{\mathrm{u},-} & \hat{\Sigma}^{\mathrm{s},-} \\ \hat{\Sigma}^{\mathrm{s},-} & \hat{\Sigma}^{\mathrm{l},-} \end{bmatrix}$$
$$= \begin{bmatrix} \left(\mathbf{I}_m - \hat{\mathbf{Q}}^{\mathrm{u}}\right)\hat{\Sigma}^{\mathrm{u},-} & \left(\mathbf{I}_m - \hat{\mathbf{Q}}^{\mathrm{u}}\right)\hat{\Sigma}^{\mathrm{s},-} \\ -\hat{\mathbf{Q}}^{\mathrm{l}}\hat{\Sigma}^{\mathrm{u},-} + \hat{\Sigma}^{\mathrm{s},-} & -\hat{\mathbf{Q}}^{\mathrm{l}}\hat{\Sigma}^{\mathrm{s},-} + \hat{\Sigma}^{\mathrm{l},-} \end{bmatrix}.$$

Hence the diagonals of the individual parts can be computed as

$$\boldsymbol{\sigma}^{\mathrm{u},+} = \left(\mathbf{1}_m - \mathbf{q}^{\mathrm{u}}\right) \odot \boldsymbol{\sigma}^{\mathrm{u},-} \tag{B.7}$$
$$\boldsymbol{\sigma}^{\mathrm{s},+} = \left(\mathbf{1}_m - \mathbf{q}^{\mathrm{u}}\right) \odot \boldsymbol{\sigma}^{\mathrm{s},-} \tag{B.8}$$
$$\boldsymbol{\sigma}^{\mathrm{l},+} = \boldsymbol{\sigma}^{\mathrm{l},-} - \mathbf{q}^{\mathrm{l}} \odot \boldsymbol{\sigma}^{\mathrm{s},-}. \tag{B.9}$$

B.2. Root Mean Square Error Results

To evaluate the actual prediction performance of our approach we repeated some experiments using the RMSE as loss function. Other than that and removing the variance output of the decoder no changes were made to the model, hyperparameters and learning procedure. The results can be found in Table B.1.

B.3. Visualization of Imputation Results

Exemplary results of the data imputation experiments conducted for the Pendulum and Quad Link experiment can be found in Figure B.1

B.4. Network Architectures and Hyper Parameters

For all experiments Adam (Kingma and Ba, 2014) with default parameters ($\alpha = 10^{-3}$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\varepsilon = 10^{-8}$) was used as an optimizer. The gradients were computed using (truncated) Backpropagation Trough Time (BpTT) (Werbos, 1990). Further in all (transposed) convolutional layers layer normalization (LN) (Ba et al., 2016) was employed to normalize the filter responses. "Same" padding was used. The elu activation function (Clevert et al., 2015) plus a constant 1 is denoted by $(\mathrm{elu} + 1)$ was used to ensure that the variance outputs are positive.

B.4.1. Pendulum and Multiple Pendulum Experiments

Observations. `Pendulum`: Grayscale images of size 24×24 pixels. `Multiple Pendulum`: RGB images of size 24×24 pixels. See Figure B.2 for examples.

Dataset. 1000 Train and 500 Test sequences of length 150. For the filtering experiments noise according to Appendix B.5 was added , for imputation 50% of the images were removed randomly.

Table B.1.: RMSE Results

Model	RMSE
Pendulum	
RKN ($m = 15, b = 3, K = 15$)	0.0779 ± 0.0082
RKN ($m = b = 15, K = 15$)	0.0758 ± 0.0094
LSTM ($m = 50$)	0.0920 ± 0.0774
LSTM ($m = 6$)	0.0959 ± 0.0100
GRU ($m = 50$)	0.0821 ± 0.0084
GRU ($m = 8$)	0.0916 ± 0.0087
Multiple Pendulums	
RKN ($m = 45, b = 3, k = 15$)	0.0878 ± 0.0036
LSTM ($m = 50$)	0.098 ± 0.0036
LSTM ($m = 12$)	0.104 ± 0.0043
GRU ($m = 50$)	0.112 ± 0.0371
GRU ($m = 14$)	0.105 ± 0.0055
Quad Link (without additional noise)	
RKN ($m = 100, b = 25, k = 15$)	0.103 ± 0.00076
LSTM ($m = 100$)	0.175 ± 0.182
LSTM ($m = 25$)	0.118 ± 0.0049
GRU ($m = 100$)	0.278 ± 0.105
GRU ($m = 25$)	0.121 ± 0.0021
Quad Link (with additional noise)	
RKN ($m = 100, b = 25, k = 15$)	0.171 ± 0.0039
LSTM ($m = 75$)	0.175 ± 0.0022
GRU ($m = 25$)	0.204 ± 0.0023

Encoder. 2 convolution + 1 fully connected + linear output & (elu + 1) output:

- Convolution 1: 12, 5×5 filter, ReLU, 2×2 max pool with 2×2 stride
- Convolution 2: 12, 3×3 filter with 2×2 stride, ReLU, 2×2 max pool with 2×2 stride
- `Pendulum`: Fully Connected 1: 30, ReLU
- `Multiple Pendulum`: Fully Connected 1: 90, ReLU

Transition Model `Pendulum`: 15 dimensional latent observation, 30 dimensional latent state. `Multiple Pendulum`: 45 dimensional latent observation, 90 dimensonal latent state. `Both`: bandwidth: 3, number of basis: 15

- $\alpha(\mathbf{z}_t)$: No hidden layers - softmax output

Decoder (for $\mathbf{s}_t^+$). 1 fully connected + linear output:

- Fully Connected 1: 10, ReLU

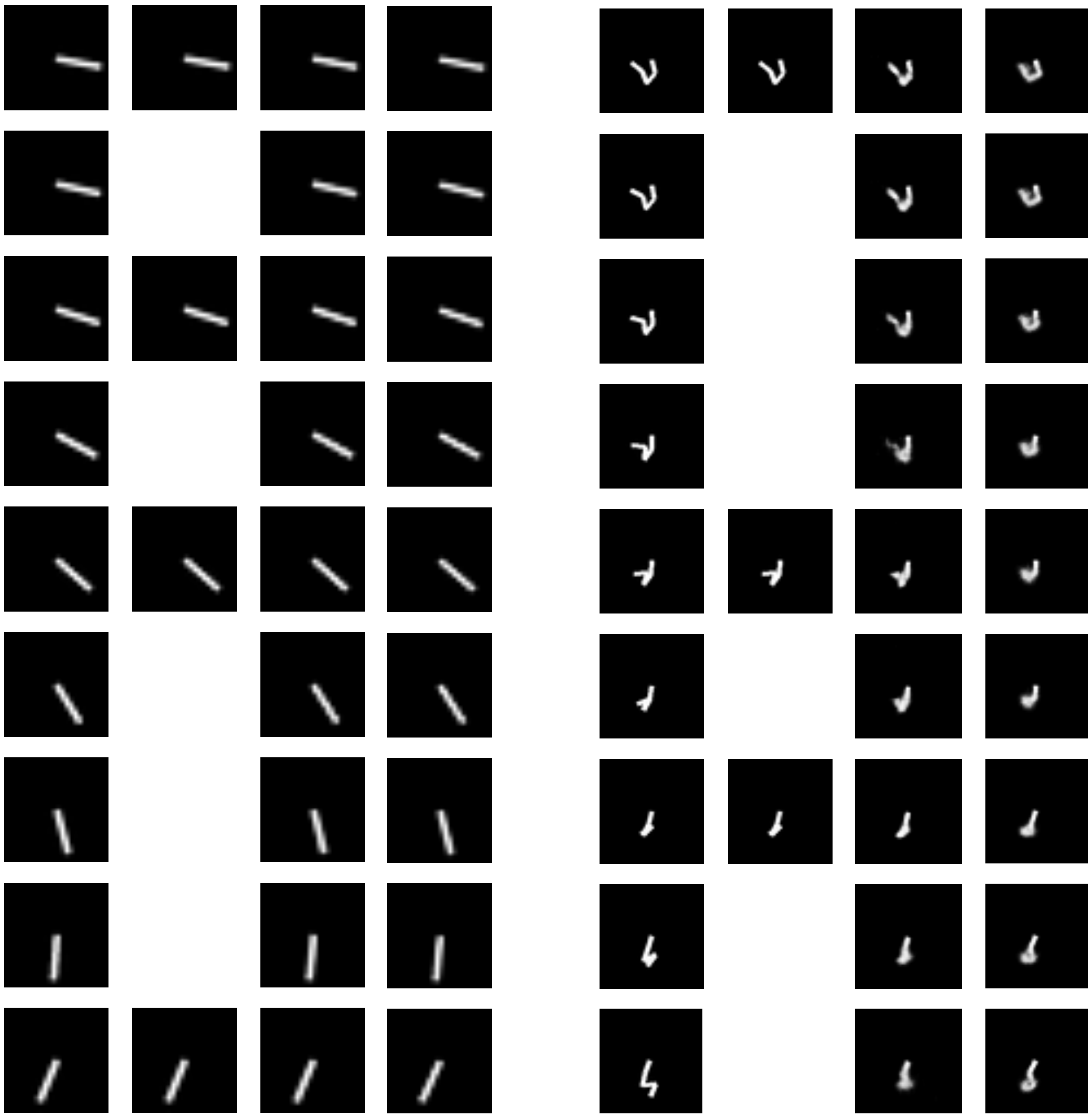


Figure B.1.: Left. Pendulum imputation Tasks. **Right.** Quad Link imputation task. Both show from left to right: true images, input to the models, imputation results for RKF, imputation results for KVAE (Fraccaro et al., 2017).

Decoder (for $\mathbf{o}_t^+$).: 1 fully connected + 2 transposed convolution + transposed convolution output:

- Fully Connected 1: 144 ReLU
- Transposed Convolution 1: 16, 5×5 filter with 4×4 stride, ReLU
- Transposed Convolution 2: 12, 3×3 filter with 2×2 stride, ReLU
- Transposed Convolution Out: `Pendulum`: 1 `Multiple Pendulum`: 3, 3×3 filter with 1×1 stride, Sigmoid

Decoder (for σ_t^+ or σ_t^+). 1 fully connected + (elu + 1):

- Fully Connected 1: 10, ReLU

B.4.2. Quad Link

Observations. Grayscale images of size $48x48$ pixels.

Dataset. 4000 Train and 1000 Test sequences of length 150. For the filtering with additional noise experiments noise according to section D was added, for imputation 50% of the images were removed randomly.

Encoder. 2 convolution + 1 fully connected + linear output & (elu + 1) output:

- Convolution 1: 12, 5×5 filter with 2×2 stride, ReLU, 2×2 max pool with 2×2 stride
- Convolution 2: 12, 3×3 filter with 2×2 stride, ReLU, 2×2 max pool with 2×2 stride
- Fully Connected 1: 200 ReLU

Transition Model. 100 dimensional latent observation, 200 dimensional latent state, bandwidth: 3, number of basis: 15

- $\alpha(\mathbf{z}_t)$: No hidden layers - softmax output

Decoder (for $\mathbf{s}_t^+$). 1 fully connected + linear output:

- Fully Connected 1: 10, ReLU

Decoder (for $\mathbf{o}_t^+$). 1 fully connected + 2 transposed convolution + transposed convolution output:

- Fully Connected 1: 144 ReLU
- Transposed Convolution 1: 16, 5×5 filter with 4×4 stride, ReLU
- Transposed Convolution 2: 12, 3×3 filter with 4×4 stride, ReLU
- Transposed Convolution Out: 1, 1×1 stride, Sigmoid

Decoder (for σ_t^+ or σ_t^+. 1 fully connected + (elu + 1):

- Fully Connected 1: 10, ReLU

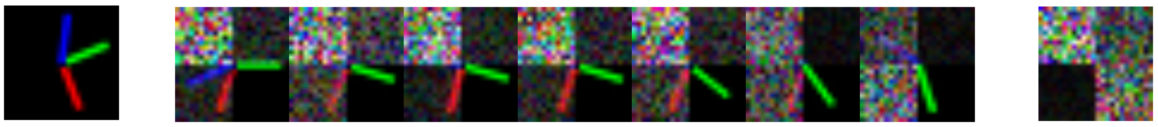


Figure B.2.: Example images for the multiple pendulum experiments. **Left**: Noise free image. **Middle**: sequence of images showing how the noise affects different pendulums differently. **Right**: Image without useful information.

B.4.3. Kitti

Observation and Data Set. For this experiment, our encoder is based on the pose network proposed by Zhou et al. (2017) which helps us to speed-up the training process. Specifically we extract features from the conv6 layer of the pose network by running the model on the KITTI odometry dataset. The training dataset for this experiment comprised of sequences 00, 01, 02, 08, 09. Sequences 03, 04, 05, 06, 07, 10 were used for testing.

Encoder. Pose Network of Zhou et al. (2017) up to layer conv6 + 1 Convolution

- Convolution 1: 50, 1x1 filter,with 1x1 stride

Transition Model. 50 dimensional latent observations, 100 dimensional latent state, bandwidth 1, number of basis 16

- $\alpha(\mathbf{z}_t)$: No hidden layers - softmax output

Decoder (for $\mathbf{s}_t^+$). 2 fully connected + linear output:

- Fully Connected 1: 50, ReLU

B.4.4. Pneumatic Brook Robot Arm

Observations and Data Set. 6 sequences of $30,000$ samples each of input currents and observed joint positions, sampled at 100Hz. 5 sequences were used for training, 1 for testing.

Encoder. 1 fully connected + linear output & $(\text{elu} + 1)$ output.

- Fully Connected, 30 ReLU

Transition Model. 30 dimensional latent observation, 60 dimensional latent state, bandwidth 3, number of basis 32

Decoder (for $\mathbf{s}_t^+$). 1 fully connected + linear output

- 30 ReLU

B.5. Observation Noise generation process

Let $\mathcal{U}(x, y)$ denote the uniform distribution from x to y.To generate the noise for the pendulum task for each sequence a sequence of factors f_t of same length was generated. To correlate the factors they were sampled as $f_0 \sim \mathcal{U}(0, 1)$ and $f_{t+1} = \min(\max(0, f_t + r_t), 1)$ with $r_t \sim \mathcal{U}(-0.2, 0.2)$. Afterwards, for each sequence two thresholds $t_1 \sim \mathcal{U}(0.0, 0.25)$ and $t_2 \sim \mathcal{U}(0.75, 1)$ were sampled. All $f_t < t_1$ were set to 0, all $f_t > t_2$ to 1 and the rest was linearly mapped to the interval $[0, 1]$. Finally, for each image $\mathbf{i}_t$ an image consisting of pure uniformly distributed noise $\mathbf{i}_t^{noise}$ was sampled and the observation computed as $\mathbf{o}_t = f_t \cdot \mathbf{i}_t + (1 - f_t) \cdot \mathbf{i}_t^{noise}$.

C. Appendix for Chapter 5

C.1. Derivations

C.1.1. Lower Bound Objective

Derivation of the general ELBO objective (Equation 5.1)

$$
\begin{aligned}
&\log p(\mathbf{o}_{\leq T}|\mathbf{a}_{\leq T}) = \log \frac{p(\mathbf{o}_{\leq T}, \mathbf{z}_{\leq T}|\mathbf{a}_{\leq T})}{p(\mathbf{z}_{\leq T}|\mathbf{o}_{\leq T}, \mathbf{a}_{\leq T})} \\
=&\mathbb{E}_{q(\mathbf{z}_{\leq T}|\mathbf{o}_{\leq T},\mathbf{a}_{\leq T})}\left[\log \frac{p(\mathbf{o}_{\leq T}|\mathbf{z}_{\leq T}, \mathbf{a}_{\leq T})p(\mathbf{z}_{\leq T}|\mathbf{a}_{\leq T})}{p(\mathbf{z}_{\leq T}|\mathbf{o}_{\leq T}, \mathbf{a}_{\leq T})} + \log \frac{q(\mathbf{z}_{\leq T}|\mathbf{o}_{\leq T}, \mathbf{a}_{\leq T})}{q(\mathbf{z}_{\leq T}|\mathbf{o}_{\leq T}, \mathbf{a}_{\leq T})}\right] \\
=&\underbrace{\mathbb{E}_{q(\mathbf{z}_{\leq T}|\mathbf{o}_{\leq T},\mathbf{a}_{\leq T})}\left[\log p(\mathbf{o}_{\leq T}|\mathbf{z}_{\leq T}, \mathbf{a}_{\leq T})\right] - \mathrm{KL}\left(q(\mathbf{z}_{\leq T}|\mathbf{o}_{\leq T}, \mathbf{a}_{\leq T})||p(\mathbf{z}_{\leq T}, |\mathbf{a}_{\leq T})\right)}_{\text{ELBO}} \\
&+ \underbrace{\mathrm{KL}\left(q(\mathbf{z}_{\leq T}|\mathbf{o}_{\leq T}, \mathbf{a}_{\leq T})||p(\mathbf{z}_{\leq T}|\mathbf{o}_{\leq T}, \mathbf{a}_{\leq T})\right)}_{\text{KL term}(\geq 0)} \\
=&\underbrace{\mathbb{E}_{q(\mathbf{z}_{\leq T}|\mathbf{o}_{\leq T},\mathbf{a}_{\leq T})}\left[\log \frac{p(\mathbf{o}_{\leq T}, \mathbf{z}_{\leq T}|\mathbf{a}_{\leq T})}{q(\mathbf{z}_{\leq T}|\mathbf{o}_{\leq T}, \mathbf{a}_{\leq T})}\right]}_{\text{ELBO}} + \underbrace{\mathrm{KL}\left(q(\mathbf{z}_{\leq T}|\mathbf{o}_{\leq T}, \mathbf{a}_{\leq T})||p(\mathbf{z}_{\leq T}|\mathbf{o}_{\leq T}, \mathbf{a}_{\leq T})\right)}_{\text{KL term}(\geq 0)}.
\end{aligned}
$$

Before we derive the decomposition of the KL term under the RSSM-assumptions (Equation 5.2) and the lower bound under the SSM-assumptions (Equation 5.4), let us restate the independence assumptions. For all approaches, the joint generative model factorizes as (cf. the graphical model in Figure 5.1)

$$p(\mathbf{o}_{\leq T}, \mathbf{z}_{\leq T}|\mathbf{a}_{\leq T}) = p(\mathbf{z}_0) \prod_{t=1}^{T} p(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{a}_{t-1}) \prod_{t=0}^{T} p(\mathbf{o}_t|\mathbf{z}_t).$$

The same independence assumptions imply the following decomposition for the conditional distribution over latent states given observations and actions $p(\mathbf{z}_{\leq T}|\mathbf{o}_{\leq T}, \mathbf{a}_{\leq T}) =$

$$p(\mathbf{z}_0|\mathbf{o}_{\leq T}, \mathbf{a}_{\leq T}) \prod_{t=1}^{T} p(\mathbf{z}_t|\mathbf{z}_{\leq t-1}, \mathbf{o}_{\leq T}, \mathbf{a}_{\leq T}) = p(\mathbf{z}_0|\mathbf{o}_{\leq T}, \mathbf{a}_{\leq T}) \prod_{t=1}^{T} p(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{o}_{\geq t}, \mathbf{a}_{\geq t-1}).$$

Thus, when using the same assumptions for the variational distribution $q(\mathbf{z}_{\leq T}|\mathbf{o}_{\leq T}, \mathbf{a}_{\leq T})$, it also factorizes as

$$q(\mathbf{z}_{\leq T}|\mathbf{o}_{\leq T}, \mathbf{a}_{\leq T}) = q(\mathbf{z}_0|\mathbf{o}_{\leq T}, \mathbf{a}_{\leq T}) \prod_{t=1}^{T} q(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{o}_{\geq t}, \mathbf{a}_{\geq t-1}).$$

Contrary to that, as introduced by (Hafner et al., 2019), the RSSM's joint variational distribution factorizes as (cf. the graphical model in Figure 5.1)

$$q(\mathbf{o}_{\leq T}, \mathbf{z}_{\leq T} | \mathbf{a}_{\leq T}) = q(\mathbf{z}_0|\mathbf{o}_0) \prod_{t=1}^{T} q(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{o}_t, \mathbf{a}_{t-1}) \prod_{t=0}^{T} q(\mathbf{o}_t).$$

Thus the variational distribution is given by

$$q(\mathbf{z}_{\leq T}|\mathbf{o}_{\leq T}, \mathbf{a}_{\leq T}) = \frac{q(\mathbf{o}_{\leq T}, \mathbf{z}_{\leq T}|\mathbf{a}_{\leq T})}{q(\mathbf{o}_{\leq T})} = q(\mathbf{z}_0|\mathbf{o}_0) \prod_{t=1}^{T} q(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{o}_t, \mathbf{a}_{t-1}).$$

Let us now consider the KL term of the lower bound decomposition (Equation 5.1) when using the generative assumptions for p and the RSSM's inference assumptions for q. We get Equation 5.2:

$$\begin{aligned}
&\mathrm{KL}\left(q(\mathbf{z}_{\leq T}|\mathbf{o}_{\leq T}, \mathbf{a}_{\leq T}) || p(\mathbf{z}_{\leq T}|\mathbf{o}_{\leq T}, \mathbf{a}_{\leq T})\right) \\
&= \int q(\mathbf{z}_0|\mathbf{o}_0) \prod_{t=1}^{T} q(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{o}_t, \mathbf{a}_{t-1}) \left(\sum_{t=1}^{T} \log \frac{q(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{o}_t, \mathbf{a}_{t-1})}{p(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{o}_{\geq t}, \mathbf{a}_{\geq t-1})} \right) d\mathbf{z}_{\leq T} \\
&= \sum_{t=1}^{T} \int \left(q(\mathbf{z}_0|\mathbf{o}_0) \prod_{t=1}^{T} q(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{o}_{\geq t}, \mathbf{a}_{\geq t-1}) \right) \log \frac{q(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{o}_t, \mathbf{a}_{t-1})}{p(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{o}_{\geq t}, \mathbf{a}_{\geq t-1})} d\mathbf{z}_{\leq T} \\
&= \sum_{t=0}^{T} \int \underbrace{\prod_{k=t+1}^{T} q(\mathbf{z}_k|\mathbf{z}_{k-1}, \mathbf{o}_k, \mathbf{a}_{k-1}) d\mathbf{z}_k}_{=1} \underbrace{q(\mathbf{z}_0|\mathbf{o}_0) \prod_{k=1}^{t-2} q(\mathbf{z}_{k+1}|\mathbf{z}_k, \mathbf{o}_{k+1}, \mathbf{a}_k) d\mathbf{z}_k}_{q(\mathbf{z}_{t-1}|\mathbf{o}_{t-1}, \mathbf{a}_{t-2})} \\
&\quad \underbrace{q(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{o}_t, \mathbf{a}_{t-1}) \log \frac{q(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{o}_t, \mathbf{a}_{t-1})}{p(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{o}_{\geq t}, \mathbf{a}_{\geq t-1})} d\mathbf{z}_t}_{\mathrm{KL}(q(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{o}_t, \mathbf{a}_{t-1})||p(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{o}_{\geq t}, \mathbf{a}_{\geq t-1}))} d\mathbf{z}_{t-1} \\
&= \sum_{t=0}^{T} \mathbb{E}_{q(\mathbf{z}_{t-1}|\mathbf{o}_{\leq t-1}, \mathbf{a}_{\leq t-2})} \left[\mathrm{KL}\left(q(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{o}_t, \mathbf{a}_{t-1}) || p(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{o}_{\geq t}, \mathbf{a}_{\geq t-1})\right) \right].
\end{aligned}$$

Plugging the same assumptions into the ELBO part of Equation 5.1 gives the RSSM objective already derived by Hafner et al. (2019) (Equation 5.3).

To derive our objective (Equation 5.4) we instead use the generative assumptions for both the generative model as well as the inference model. Again starting from the ELBO part of Equation 5.1, we get

$$\begin{aligned}
&\mathbb{E}_{q(\mathbf{z}_{\leq T}|\mathbf{o}_{\leq T}, \mathbf{a}_{\leq T})} \left[\log p(\mathbf{o}_{\leq T}|\mathbf{z}_{\leq T}) \right] - \mathrm{KL}\left(q(\mathbf{z}_{\leq T}|\mathbf{o}_{\leq T}, \mathbf{a}_{\leq T}) || p(\mathbf{z}_{\leq T}|\mathbf{a}_{\leq T})\right) \\
&= \int \left(q(\mathbf{z}_0|\mathbf{o}_{\leq T}, \mathbf{a}_{\leq T}) \prod_{t=1}^{T} q(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{a}_{\geq t-1}, \mathbf{o}_{\geq t}) \right) \sum_{t=0}^{T} \log p(\mathbf{o}_t|\mathbf{z}_t) d\mathbf{z}_{\leq t} \\
&\quad - \int q(\mathbf{z}_0|\mathbf{o}_{\leq T}, \mathbf{a}_{\leq T}) \prod_{t=1}^{T} q(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{a}_{\geq t-1}, \mathbf{o}_{\geq t}) \left(\sum_{t=1}^{T} \log \frac{q(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{a}_{\geq t-1}, \mathbf{o}_{\geq t})}{p(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{a}_{t-1})} \right) d\mathbf{z}_{\leq T} \\
&= \sum_{t=0}^{T} \int \left(q(\mathbf{z}_0|\mathbf{o}_{\leq T}, \mathbf{a}_{\leq T}) \prod_{t=1}^{T} q(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{a}_{\geq t-1}, \mathbf{o}_{\geq t}) \right) \log p(\mathbf{o}_t|\mathbf{z}_t) d\mathbf{z}_{\leq t}
\end{aligned}$$

$$
\begin{aligned}
&-\sum_{t=1}^{T}\int\left(q(\mathbf{z}_0|\mathbf{o}_{\leq T},\mathbf{a}_{\leq T})\prod_{t=1}^{T}q(\mathbf{z}_t|\mathbf{z}_{t-1},\mathbf{a}_{\geq t-1},\mathbf{o}_{\geq t}\right)\log\frac{q(\mathbf{z}_t|\mathbf{z}_{t-1},\mathbf{a}_{\geq t-1},\mathbf{o}_{\geq t})}{p(\mathbf{z}_t|\mathbf{z}_{t-1},\mathbf{a}_{t-1})}d\mathbf{z}_{\leq T}\\
=&\sum_{t=0}^{T}\int\underbrace{\prod_{k=t+1}^{T}q(\mathbf{z}_k|\mathbf{z}_{k-1},\mathbf{a}_{\geq k-1},\mathbf{o}_{\geq k})d\mathbf{z}_k}_{=1}\\
&\quad\underbrace{q(\mathbf{z}_0|\mathbf{o}_{\leq T},\mathbf{a}_{\leq T})\prod_{k=1}^{t-1}q(\mathbf{z}_{k+1}|\mathbf{z}_k,\mathbf{a}_{\geq k},\mathbf{o}_{\geq k+1})d\mathbf{z}_k}_{q(\mathbf{z}_t|\mathbf{o}_{\leq T},\mathbf{a}_{\leq T})}\log p(\mathbf{o}_t|\mathbf{z}_t)d\mathbf{z}_t\\
&-\sum_{t=0}^{T}\int\underbrace{\prod_{k=t+1}^{T}q(\mathbf{z}_k|\mathbf{z}_{k-1},\mathbf{a}_{\geq k-1},\mathbf{o}_{\geq k})d\mathbf{z}_k}_{=1}\underbrace{(q(\mathbf{z}_0|\mathbf{o}_{\leq T},\mathbf{a}_{\leq T})\prod_{k=1}^{t-2}q(\mathbf{z}_{k+1}|\mathbf{z}_k,\mathbf{a}_{\geq k},\mathbf{o}_{\geq k+1})d\mathbf{z}_k}_{q(\mathbf{z}_{t-1}|\mathbf{o}_{\leq T},\mathbf{a}_{\leq T})}\\
&\quad\underbrace{q(\mathbf{z}_t|\mathbf{z}_{t-1},\mathbf{a}_{\geq t-1},\mathbf{o}_{\geq t},)\log\frac{q(\mathbf{z}_t|\mathbf{z}_{t-1},\mathbf{a}_{\geq t-1},\mathbf{o}_{\geq t})}{p(\mathbf{z}_t|\mathbf{z}_{t-1})}d\mathbf{z}_t}_{\mathrm{KL}(q(\mathbf{z}_t|\mathbf{z}_{t-1},\mathbf{a}_{\geq t-1},\mathbf{o}_{\geq t})||p(\mathbf{z}_t|\mathbf{z}_{t-1},\mathbf{a}_{t-1}))}d\mathbf{z}_{t-1}\\
=&\sum_{t=0}^{T}\mathbb{E}_{q(\mathbf{z}_t|\mathbf{o}_{\leq T},\mathbf{a}_{\leq T})}\left[\log p(\mathbf{o}_t|\mathbf{z}_t)\right]\\
&-\sum_{t=1}^{T}\mathbb{E}_{q(\mathbf{z}_{t-1}|\mathbf{o}_{\leq T},\mathbf{a}_{\leq T})}\left[\mathrm{KL}\left(q(\mathbf{z}_t|\mathbf{z}_{t-1},\mathbf{a}_{\geq t-1},\mathbf{o}_{\geq t})||p(\mathbf{z}_t|\mathbf{z}_{t-1},\mathbf{a}_{t-1})\right)\right].
\end{aligned}
$$

C.1.2. Extended Backward Pass for Smoothed Dynamics

Here we derive an extended backward pass, i.e., given the forward priors and posteriors we are interested in the smoothed state $q(\mathbf{z}_{t-1}|\mathbf{o}_{\leq T},\mathbf{a}_{\leq T})$ as well as the smoothed dynamics $q(\mathbf{z}_t|\mathbf{z}_{t-1},\mathbf{a}_{\geq t-1},\mathbf{o}_{\geq t})$. We can read off both these quantities from the joint smoothed estimate

$$q(\mathbf{z}_t,\mathbf{z}_{t-1}|\mathbf{o}_{\leq T},\mathbf{a}_{\leq T})=q(\mathbf{z}_t|\mathbf{z}_{t-1},\mathbf{a}_{\geq t-1},\mathbf{o}_{\geq t})q(\mathbf{z}_{t-1}|\mathbf{o}_{\leq T},\mathbf{a}_{\leq T}).$$

This joint smoothed estimate can be computed using the identity

$$
\begin{aligned}
q(\mathbf{z}_t,\mathbf{z}_{t-1}|\mathbf{o}_{\leq T},\mathbf{a}_{\leq T})&=q(\mathbf{z}_{t-1}|\mathbf{z}_t,\mathbf{o}_{\leq t-1},\mathbf{a}_{\leq t-1})q(\mathbf{z}_t|\mathbf{o}_{\leq T},\mathbf{a}_{\leq T})\\
=&\frac{p(\mathbf{z}_t|\mathbf{z}_{t-1},\mathbf{a}_{t-1})q(\mathbf{z}_{t-1}|\mathbf{o}_{\leq t-1},\mathbf{a}_{\leq t-2})}{q(\mathbf{z}_t|\mathbf{o}_{\leq t-1},\mathbf{a}_{\leq t-1})}q(\mathbf{z}_t|\mathbf{o}_{\leq T},\mathbf{a}_{\leq T}).
\end{aligned}\tag{C.1}
$$

All quantities in Equation C.1 are available in closed form as the smoothed state-estimate $q(\mathbf{z}_t|\mathbf{o}_{\leq T},\mathbf{a}_{\leq T})$ is computed during the previous iteration of the backward pass. To recursion starts with the last posterior which is equal to the smoothed estimate for $t=T$. We denote the parameters of the 4 distributions as follows

$$
\begin{aligned}
&q(\mathbf{z}_t|\mathbf{o}_{\leq t-1},\mathbf{a}_{\leq t-1})=\mathcal{N}(\mathbf{z}_t|\boldsymbol{\mu}_t^-,\Sigma_t^-),\quad q(\mathbf{z}_{t-1}|\mathbf{o}_{\leq t-1},\mathbf{a}_{\leq t-2})=\mathcal{N}(\mathbf{z}_{t-1}|\boldsymbol{\mu}_{t-1}^+,\Sigma_{t-1}^+),\\
&q(\mathbf{z}_t|\mathbf{o}_{\leq T},\mathbf{a}_{\leq T})=\mathcal{N}(\mathbf{z}_t|\boldsymbol{\mu}_t^{\mathrm{s}},\Sigma_t^{\mathrm{s}})\ \text{ and }\ p(\mathbf{z}_t|\mathbf{z}_{t-1},\mathbf{a}_{t-1})=\mathcal{N}(\mathbf{z}_t|\mathbf{A}_{t-1}\mathbf{z}_{t-1}+\mathbf{B}_{t-1}\mathbf{a}_{t-1},\Sigma_{t-1}^{\mathrm{dyn}}).
\end{aligned}
$$

Using Bayes rule we get

$$q(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{o}_{\leq t-1}, \mathbf{a}_{\leq t-1}) = \mathcal{N}(\boldsymbol{\mu}^+_{t-1} + \mathbf{C}_{t-1}(\mathbf{z}_t - \underbrace{\mathbf{A}_{t-1}\boldsymbol{\mu}^+_{t-1} + \mathbf{B}_{t-1}\mathbf{a}_{t-1}}_{\boldsymbol{\mu}^-_t}), (\mathbf{I} - \mathbf{C}_{t-1}\mathbf{A}_{t-1})\Sigma^+_{t-1})$$

with $\mathbf{C}_{t-1} = \Sigma^+_{t-1}\mathbf{A}^T_{t-1}\underbrace{(\mathbf{A}_{t-1}\Sigma^+_{t-1}\mathbf{A}^T_{t-1} + \Sigma^{\text{dyn}}_{t-1})}_{\Sigma^-_t}{}^{-1} = \Sigma^+_{t-1}\mathbf{A}^T_{t-1}\left(\Sigma^-_t\right)^{-1}$. Thus, the full "two-sliced" smoothed estimate is given by

$$q(\mathbf{z}_{t-1}, \mathbf{z}_t|\mathbf{o}_{\leq T}, \mathbf{a}_{\leq T}) = q(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{o}_{\leq t-1}, \mathbf{a}_{\leq t-1})q(\mathbf{z}_t|\mathbf{o}_{\leq T}, \mathbf{a}_{\leq T}) =$$
$$\mathcal{N}\left(\begin{pmatrix}\mathbf{z}_{t-1}\\ \mathbf{z}_t\end{pmatrix} | \begin{pmatrix}\boldsymbol{\mu}^+_{t-1} + \mathbf{C}_{t-1}(\boldsymbol{\mu}^s_t - \boldsymbol{\mu}^-_t)\\ \boldsymbol{\mu}^s_t\end{pmatrix}, \begin{pmatrix}(\mathbf{I} - \mathbf{C}_{t-1}\mathbf{A}_{t-1})\Sigma^+_{t-1} + \mathbf{C}_{t-1}\Sigma^s_t\mathbf{C}^T_{t-1} & \mathbf{C}_{t-1}\Sigma^s_t\\ \Sigma^s_t\mathbf{C}^T_{t-1} & \Sigma^s_t\end{pmatrix}\right)$$
$$= \mathcal{N}\left(\begin{pmatrix}\mathbf{z}_{t-1}\\ \mathbf{z}_t\end{pmatrix} | \begin{pmatrix}\boldsymbol{\mu}^s_{t-1}\\ \boldsymbol{\mu}^s_t\end{pmatrix}, \begin{pmatrix}\Sigma^s_{t-1} & \mathbf{C}_{t-1}\Sigma^s_t\\ \Sigma^s_t\mathbf{C}^T_{t-1} & \Sigma^s_t\end{pmatrix}\right).$$

We can just read off the smoothed dynamics at time t from the joint above and get

$$q(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{a}_{\geq t-1}, \mathbf{o}_{\geq t})$$
$$= \mathcal{N}\left(\mathbf{z}_t|\boldsymbol{\mu}^s_t + \Sigma^s_t\mathbf{C}^T_{t-1}(\Sigma^s_{t-1})^{-1}(\mathbf{z}_{t-1} - \boldsymbol{\mu}^s_{t-1}), \Sigma^s_t - \Sigma^s_t\mathbf{C}^T_{t-1}(\Sigma^s_{t-1})^{-1}\mathbf{C}_{t-1}\Sigma^s_t\right).$$

We can also read off the smoothed state estimate at time $t-1$ and get the standard Rauch-Tung-Striebel backward recursion.

C.2. Details on Experiment Setup, Baselines and Hyperparameters

We evaluate all variants of the RSSM and the VRKN in an experimental setup that closely follows Hafner et al. (2019) as well as Hafner et al. (2020) by exchanging the underlying model while keeping all other parts of the approach fix. We refer to the pseudo-code presented in those works for an overview of the training procedure.

C.2.1. Smoothing RSSM

The smoothing RSSM works very similar to the $\mathcal{L}_{\text{ssm}}(\text{CF})$ and uses a GRU (Cho et al., 2014) to accumulate information for all future time steps. This GRU is placed between the CNN which extracts the representation for the RSSM and the RSSM itself, as indicated by the dashed red line in Figure C.1. It processes the CNN's output backward and extracts a latent representation of all future observations $\mathbf{o}_{\geq t}$ which is fed to the rest of the pipeline instead of the original CNN output. For online control, we only have $\mathbf{o}_t$ as all future information is unavailable. In this case, we only feed $\mathbf{o}_t$, together with the default initial memory value, through the GRU and give the output to the remaining pipeline.

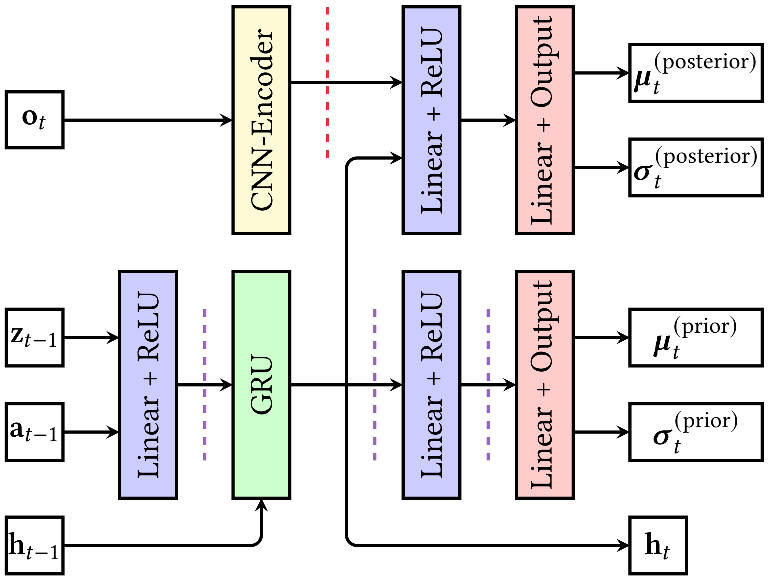


Figure C.1.: Overview of the dynamics and inference model of the RSSM with MC Dropout baseline. The general architecture is equivalent to that of the original RSSM, but we included Monte Carlo Dropout Layers at the locations indicated by the purple dashed lines. For preliminary experiments, we also included Dropout before the two linear layers in the computation path of the posterior but found that this decreases performance. The dashed red line indicates the position where the backward GRU is added for the smoothing version of the RSSM

.

C.2.2. RSSM with Monte Carlo Dropout

For the baseline with Bayesian treatment of the RSSM's transition parameters, we combined the original RSSM with Monte Carlo Dropout (Gal and Ghahramani, 2016). Figure C.1 provides an overview over the resulting architecture.

C.2.3. Missing Observations and Fusion for RSSM.

For the experiments including missing observations, we concatenated the flag indicating the validity of the observation after the CNN-Encoder in Figure C.1.

For fusion, we used several encoders, whose output we concatenate before passing it to the next linear layer (red dashed line in Figure C.1). Like for the VRKN, we use a separate decoder an reconstruction loss term for all observations (Section 5.3.6)

C.2.4. Proprioceptive Joints in Fusion Tasks

Table C.1 shows how we split the observations in into proprioceptive and non-proprioceptive information for the different environments. In all cases, we only provide positional information, no velocities.

Environment	Proprioceptive	Non-Proprioceptive
`Cheetah Run`	Joints (`bthigh, bshin, bfoot, fthigh, fshin, ffoot`)	Global Position/Orientation: (`rootx, rooty, rootz`)
`Walker Walk / Run`	orientation of links	height of center of mass above the ground
`Cartpole Swingup`	Cart position (`slider`)	Pole angle (`hinge`)
`Cup Catch`	Cup Position (`cup_x, cup_z`)	Ball Position(`ball_x, ball_z`)

Table C.1.: For the fusion experiment we split the standard observations of the environments into proprioceptive and non-proprioceptive as follows.

C.2.5. Hyperparameters

Where applicable we reused the hyperparametes used in (Hafner et al., 2019) and (Hafner et al., 2020).

State Space Models. The exact parametrization of the RSSM (layer widths and activation functions) differs between (Hafner et al., 2019) and (Hafner et al., 2020) and we used the respective values for the respective agents. Both use a stochastic state size of 30 and a deterministic state size of 200.

For the VRKN, we used a common set of hyper-parameters for all experiments and a stochastic state size of 230, to match the total state size of the RSSM approaches. All layers, including the GRU cell, are of width 300 and all linear layers use ReLU activation. The transition matrix output of the transition network is transformed by the sigmoid-based transformation described in Section 5.3, the offset output is unconstrained and the transition noise output is constrained by the sigmoid-based transformation, saturating at $0.001, 0.1$ while $f(0) = 0.01$.

For the Monte Carlo dropout-based approaches, we use a dropout rate of 0.1.

Encoder - Decoder. We used the convectional architectures originally introduced in (Ha and Schmidhuber, 2018) and later used in (Hafner et al., 2019, 2020). For the VRKN the encoder was augmented with a linear layer mapping to the latent observation and a linear layer + softplus for the uncertainty output.

For the reward decoder, we also followed (Hafner et al., 2019) and (Hafner et al., 2020) and used 2 layers of width 200 with ReLU activation for PlaNet experiments and 3 layers of width 300 with ELU activation for all the Dreamer experiments.

To encode low-dimensional proprioceptive observation in the fusion experiments we used an additional encoder network with 3 layers of width 300 with ELU activation.

Loss. Following the official implementation of (Hafner et al., 2019) we scaled the loss of the reward decoder by a factor of 10 for the PlaNet experiments. Ee also found this beneficial for the VRKN. While the RSSM (and all approaches based on the original parametrization) use a factor of 1 for Dreamer, we kept the factor of 10 for the VRKN-parametrization based approaches. In both cases, that led to better performance of the respective approaches. We used the 3-free nats trick used in (Hafner et al., 2019, 2020) to regularize the KL term for all experiments and approaches.

Training. We started with 5 randomly collected initial episodes and collected one more episode every 100 model update steps. Each update step is performed on 50 sub-sequences of length 50, sampled uniformly from all previously collected sequences. We used Adam (Kingma and Ba, 2014) with the same learning rates and gradient clipping as (Hafner et al., 2019) for the PlaNet experiments (learning rate 10^{-3}, clip by norm $1,000$) and (Hafner et al., 2020) for Dreamer (learning rate world model: $6 \cdot 10^{-4}$, learning rate actor and critic: $8 \cdot 10^{-5}$, all with clip by norm 100).

Environments For PlaNet-agents we used the same action repeats as (Hafner et al., 2019) (Cheetah Run: 4, Walker Walk: 2, Cup Catch: 4, Cartpole Swingup: 8, Reacher Easy: 4, Finger Spin: 2). For Dreamer-agents we used an action repeat of 2 for all environments. Images are $[64 \times 64]$ pixels and the color-depth is sub-sampled to 5-bits, following (Hafner et al., 2019). Table C.1 shows splits into proprioceptive and non-proprioceptive state information for the fusion experiments.

Data Collection and Evaluation. We collected data with an exploration noise of 0.3. For the Monte-Carlo Dropout based approaches, we turned the dropout layers off during evaluation but not during data collection.

Planet-Agent The CEM-Planning method is equivalent to the one used in (Hafner et al., 2019) for all experiments.

Dreamer-Agent Actor and value networks, as well as the hyper-parameters for the generalized value estimates, are equivalent to the one used in (Hafner et al., 2020) for all approaches up to the activation function used in the value network. Here we use Tanh to avoid rare cases of exploding values for the VRKN approaches. In the cases where the original formulation with ELU converged, we observed no difference from Tanh.

C.3. Comprehensive Experiment Results and Visualizations

We provide results for the individual tasks for all conducted experiments. Here we report reward curves, as well as the final performance. Following the suggestions of Agarwal et al. (2021), for the reward curves, we report the interquartile mean and 95% bootstrapped confidence intervals (shaded areas). Unless otherwise noted, we use 10 seeds per task and approximate the expected return using 10 rollouts. We provide box plots and tables stating the mean and standard error of the final performance. We define final performance as the mean performance during the last $100,000$ train steps.

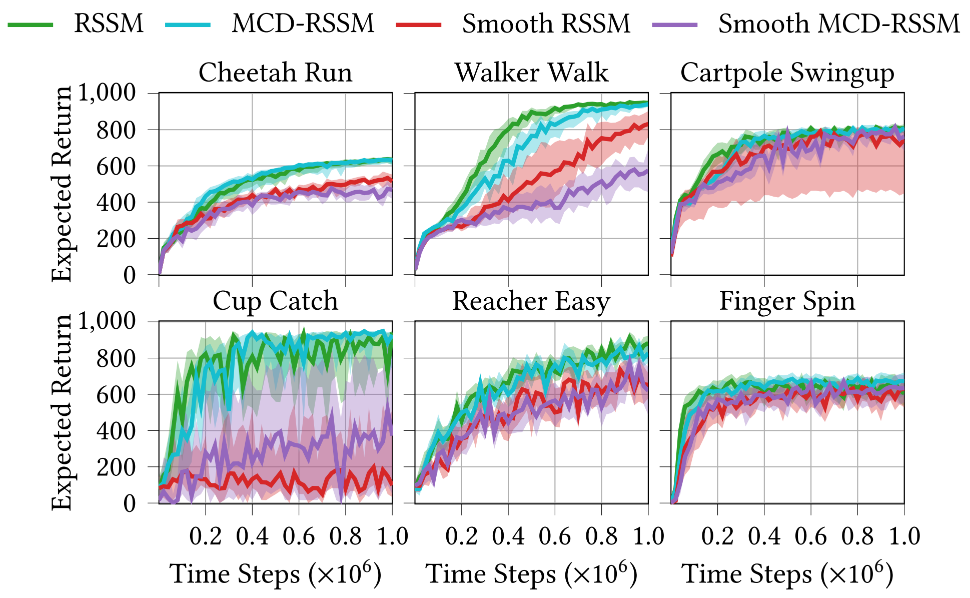


Figure C.2.: Comparison of RSSM and all baselines on all tasks considered for the PlaNet agents.

	RSSM	MCD-RSSM	Smooth RSSM	Smooth MCD-RSSM
Cheetah Run	627.4290 ± 7.4144	616.2516 ± 13.5786	519.9207 ± 11.7773	454.1092 ± 14.7283
Walker Walk	945.8944 ± 2.1465	912.0173 ± 19.3413	794.5836 ± 38.2716	557.6727 ± 33.9011
Cartpole Swingup	790.4359 ± 8.4232	791.2157 ± 8.4791	640.5327 ± 79.7364	761.4190 ± 10.7277
Cup Catch	783.3987 ± 54.1554	874.4680 ± 23.6101	233.3460 ± 78.0361	421.9260 ± 122.9805
Reacher Easy	862.8973 ± 8.1286	818.6773 ± 10.3870	654.9960 ± 23.2955	681.2360 ± 30.5072
Finger Spin	623.6027 ± 21.5528	635.7293 ± 32.7763	599.2760 ± 17.6737	622.0860 ± 19.4163

Table C.2.: Mean and standard error of final performance for the comparison of RSSM and all baselines on all tasks considered for the PlaNet agents.

We want to emphasize that we draw our conclusion and claims from the aggregated results reported in the main part of the paper.

C.3.1. Comprehensive Results for the Comparison of the Different RSSM Versions in Section 5.4.1 (Figure 5.4)

Figure C.2 shows the comparison to the Smoothing RSSM and the MCD versions for PlaNet agents. Figure C.3 and Table C.2 show the corresponding box plots and table. Figure C.4 shows the comparison to the Smoothing RSSM and the MCD versions for Dreamer agents. Figure C.5 and Table C.3 show the corresponding box plots and table.

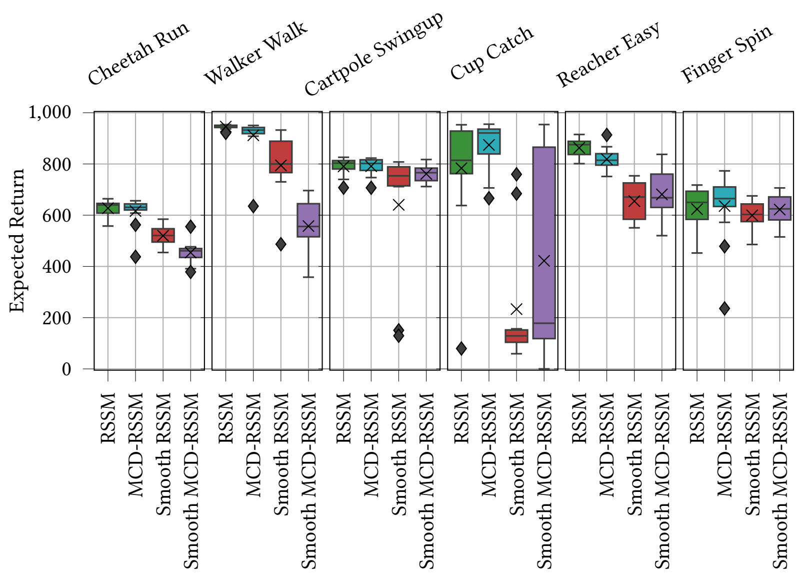


Figure C.3.: Box plots of final performance for the comparison of RSSM and all baseline on all tasks considered for the PlaNet agents.

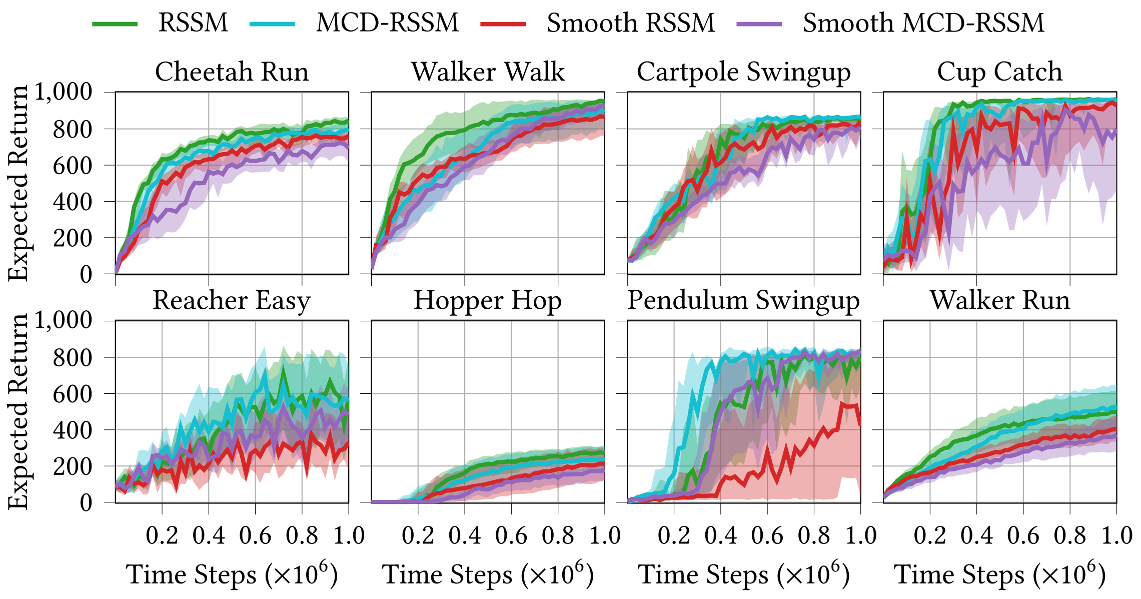


Figure C.4.: Comparison of RSSM and all baselines on all tasks considered for the Dreamer agents.

C.3.2. Comprehensive Results for VRKN and RSSM Comparison in Section 5.4.1 (Figure 5.5)

Figure C.6 shows the reward curves for PlaNet agents, Figure C.7 and Table C.4 show the corresponding box-plots and table. Figure C.8 shows the reward curves for Dreamer agents, Figure C.9 and Table C.5 show the corresponding box-plots and table.

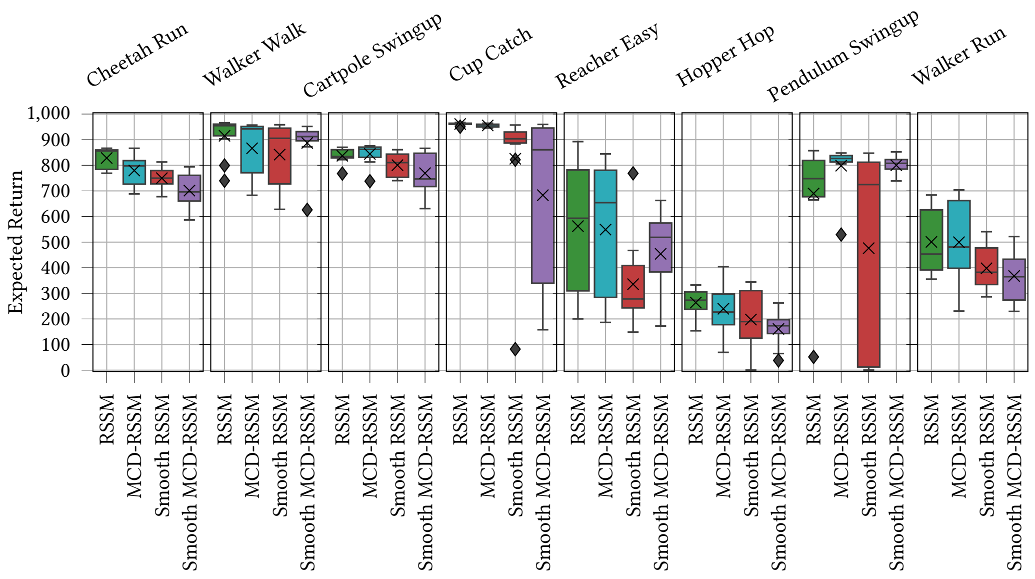


Figure C.5.: Box plots of final performance for the comparison of RSSM and all baseline on all tasks considered for the Dreamer agents.

	RSSM	MCD-RSSM	Smooth RSSM	Smooth MCD-RSSM
Cheetah Run	828.3105 ± 12.7068	779.5831 ± 17.5038	749.6598 ± 12.6213	701.1207 ± 20.8974
Walker Walk	915.0295 ± 24.0175	866.2743 ± 33.1511	841.6051 ± 37.6502	887.1364 ± 28.4298
Cartpole Swingup	837.0084 ± 8.9790	844.2569 ± 12.9288	800.4897 ± 14.5373	769.1028 ± 23.8682
Cup Catch	961.1700 ± 1.3949	955.5520 ± 2.1011	825.4520 ± 79.2088	683.2780 ± 108.1605
Reacher Easy	562.4000 ± 80.4134	548.8700 ± 82.5122	335.7420 ± 55.2733	454.1340 ± 51.4296
Hopper Hop	263.8984 ± 17.7745	239.7520 ± 29.6752	197.5490 ± 36.8571	160.7789 ± 20.0703
Pendulum Swingup	689.7720 ± 70.8294	798.3480 ± 28.6680	476.3280 ± 121.9812	799.9800 ± 10.4065
Walker Run	500.7626 ± 39.3808	499.6024 ± 52.2246	397.7456 ± 26.9991	367.6974 ± 32.9808

Table C.3.: Mean and standard error of final performance for the comparison of RSSM and all baselines on all tasks considered for the Dreamer agents.

C.3.3. Comprehensive Results for Occlusion Experiments in Section 5.4.2.1 (Figure 5.7)

Figure C.10 shows the reward curves for Dreamer agents on the disc occlusion task, Figure C.11 and Table C.6 show the corresponding box-plots and table.

Figure C.12 shows the reward curves for Dreamer agents on the wall occlusion task, Figure C.13 and Table C.7 show the corresponding box-plots and table.

C.3.4. Comprehensive Results for Missing Observation Experiments in Section 5.4.2.2 (Figure 5.9)

Figure C.14 shows the reward curves for Dreamer agents on the missing observation task without transition noise, Figure C.15 and Table C.8 show the corresponding box-plots and table.

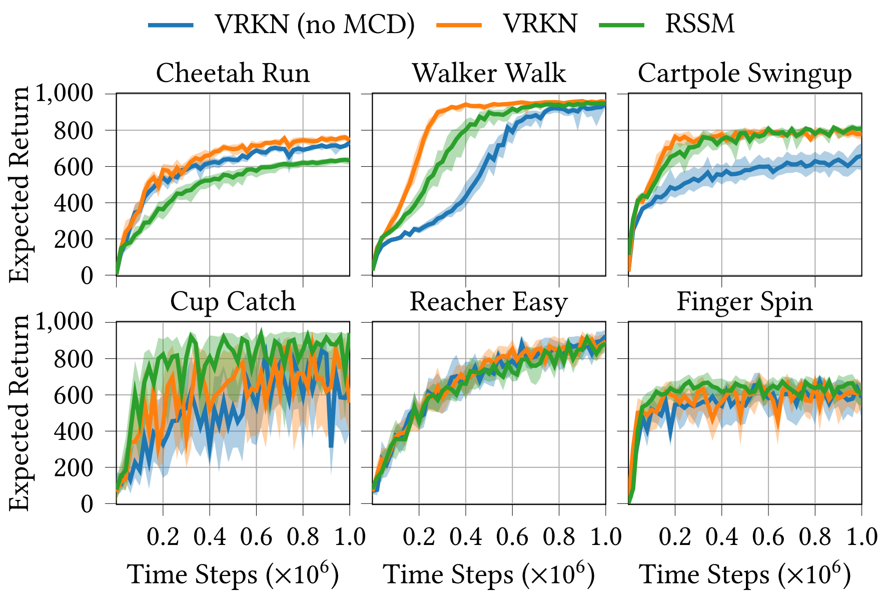


Figure C.6.: Comparison of VRKN (with and without Monte Carlo Dropout) and RSSM on all tasks considered for the PlaNet agents.

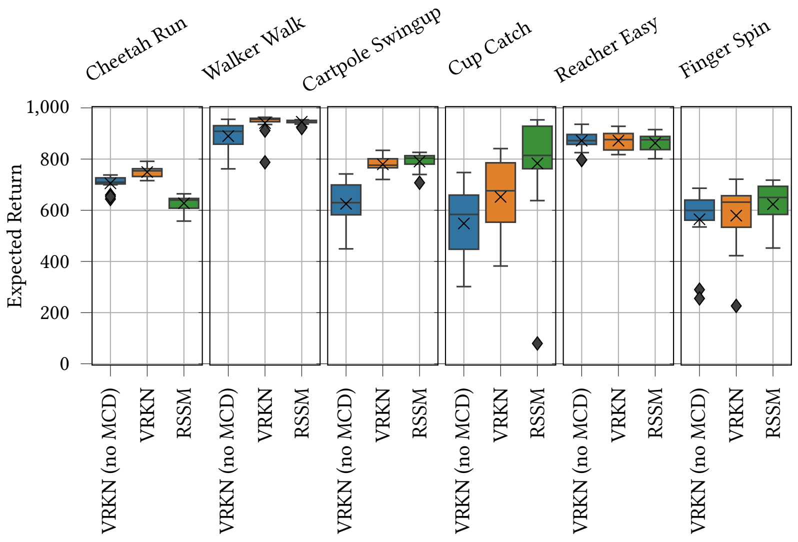


Figure C.7.: Box plots of final performance for the comparison of VRKN (with and without Monte Carlo Dropout) and RSSM on all tasks considered for the PlaNet agents.

Figure C.16 shows the reward curves for Dreamer-agents on the missing observation task with transition noise, Figure C.17 and Table C.9 show the corresponding box-plots and table.

	VRKN (no MCD)	VRKN	RSSM
Cheetah Run	705.1629 ± 7.3881	749.2355 ± 5.3298	627.4290 ± 7.4144
Walker Walk	889.8291 ± 14.2379	940.5245 ± 11.0899	945.8944 ± 2.1465
Cartpole Swingup	625.0746 ± 22.7305	779.3265 ± 7.7148	790.4359 ± 8.4232
Cup Catch	548.4413 ± 35.6197	652.2427 ± 37.5175	783.3987 ± 54.1554
Reacher Easy	871.8760 ± 10.4746	872.0493 ± 9.5175	862.8973 ± 8.1286
Finger Spin	564.7493 ± 31.3898	578.9867 ± 31.7433	623.6027 ± 21.5528

Table C.4.: Mean and standard-error of final performance for the comparison of VRKN (with and without Monte Carlo Dropout) and RSSM on all tasks considered for the PlaNet agents.

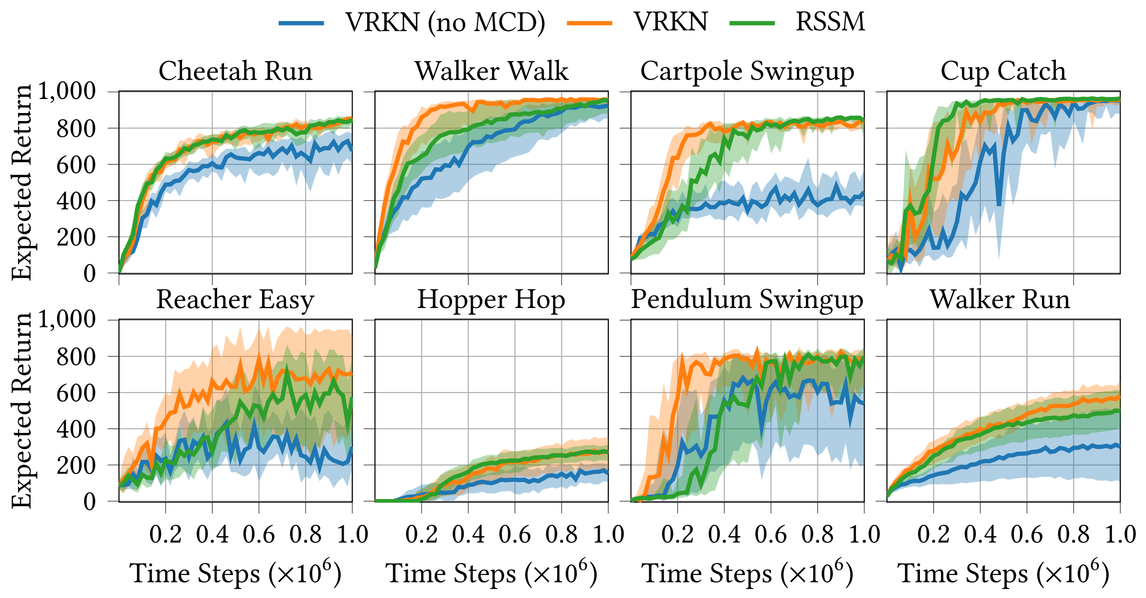


Figure C.8.: Comparison of VRKN (with and without Monte Carlo Dropout) and RSSM on all tasks considered for the Dreamer agents.

C.3.5. Comprehensive Results for Sensor Fusion Experiments in Section 5.4.2.3 (Figure 5.10)

Figure C.18 shows the reward curves for Dreamer agents on the fusion task without transition noise, Figure C.19 and Table C.10 show the corresponding box-plots and table.

Figure C.20 shows the reward curves for Dreamer agents on the fusion task with transition noise, Figure C.21 and Table C.11 show the corresponding box-plots and table.

C.3.6. Qualitative Image Reconstructions for Noisy-Dreamer Experiments

We show sub-sequences of decoded posterior belief states for the 4 noisy tasks in Figure C.22, Figure C.23, Figure C.24, and Figure C.25. Note that the first image in those sequences is not necessary the first image the respective model saw, thus the reconstruction might be reasonable even if the system is fully occluded. Figure C.26 explicitly shows the models' behavior if no information is available at the beginning of a sequence.

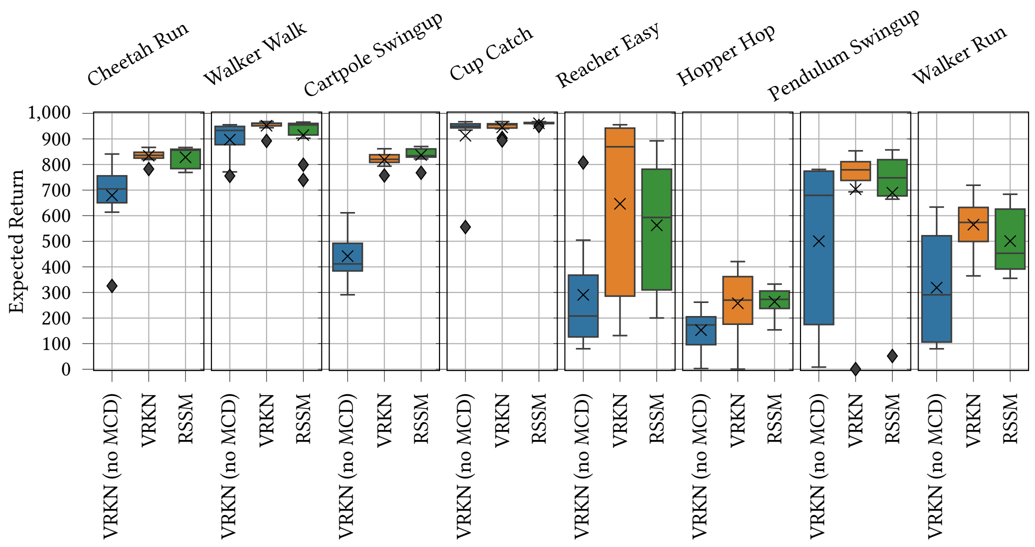


Figure C.9.: Box plots of final performance for the comparison of VRKN (with and without Monte Carlo Dropout) and RSSM on all tasks considered for the Dreamer agents.

	VRKN (no MCD)	VRKN	RSSM
Cheetah Run	678.8993 ± 42.6969	834.1799 ± 7.0240	828.3105 ± 12.7068
Walker Walk	897.1648 ± 22.6760	951.5513 ± 6.6140	915.0295 ± 24.0175
Cartpole Swingup	441.9785 ± 31.5798	818.7790 ± 8.8191	837.0084 ± 8.9790
Cup Catch	912.3980 ± 37.7156	945.1860 ± 7.7854	961.1700 ± 1.3949
Reacher Easy	290.6880 ± 68.4338	646.3940 ± 106.9958	562.4000 ± 80.4134
Hopper Hop	152.6910 ± 23.3830	257.7012 ± 38.3835	263.8984 ± 17.7745
Pendulum Swingup	500.4280 ± 98.7629	703.5000 ± 75.4019	689.7720 ± 70.8294
Walker Run	318.8323 ± 68.5679	565.1005 ± 32.1744	500.7626 ± 39.3808

Table C.5.: Mean and standard-error of final performance corresponding for the comparison of VRKN (with and without Monte Carlo Dropout) and RSSM on all tasks considered for the Dreamer agents.

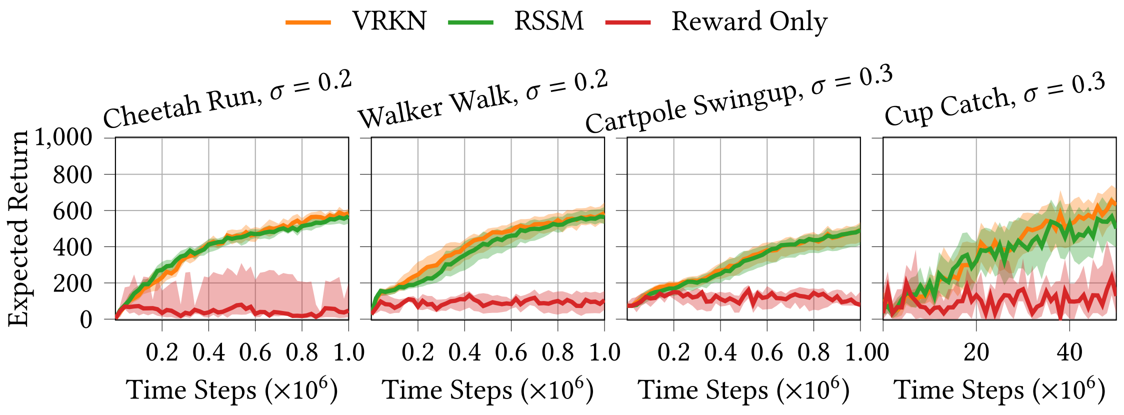


Figure C.10.: Comparison of VRKN and RSSM on all tasks considered with disc occlusions. The σ in the title indicates the standard deviation of the Gaussian transition noise.

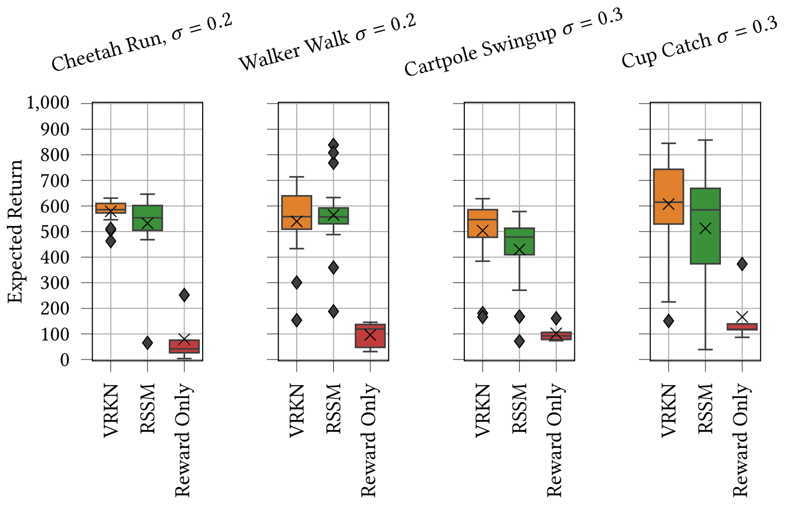


Figure C.11.: Box plots of final performance for the comparison of VRKN and RSSM on all tasks for the disc occlusion task.

	VRKN	RSSM	Reward Only
Cheetah Run	578.7090 ± 9.4257	532.2949 ± 26.3772	79.7795 ± 39.9028
Walker Walk	539.5958 ± 28.6619	564.8669 ± 30.8898	95.9638 ± 21.2174
Cartpole Swingup	503.7676 ± 28.6705	429.8483 ± 31.4116	102.1589 ± 14.0952
Cup Catch	607.3370 ± 40.2025	512.6090 ± 48.2354	166.6440 ± 46.7917

Table C.6.: Mean and standard-error of final performance for the comparison of VRKN and RSSM on all tasks for the disc occlusion task.

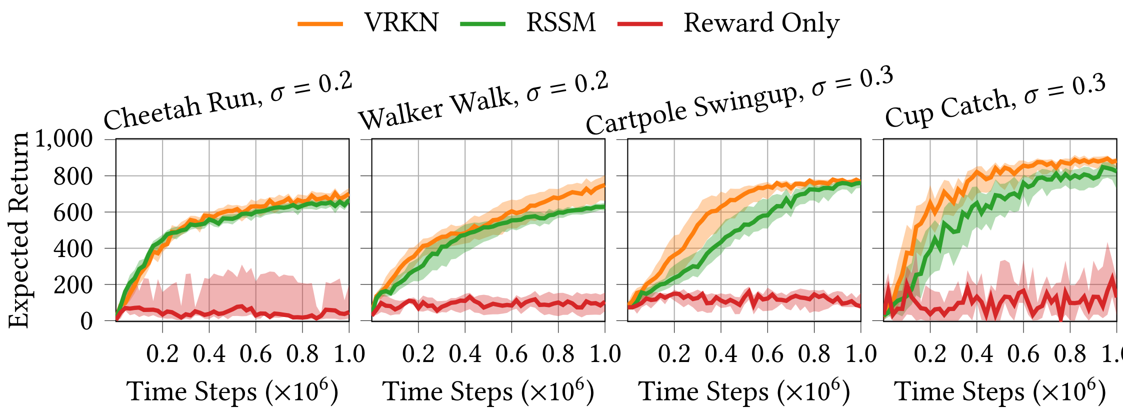


Figure C.12.: Comparison of VRKN and RSSM on all tasks considered with wall occlusions. The σ in the title indicates the standard deviation of the Gaussian transition noise.

	VRKN	RSSM	Reward Only
Cheetah Run	677.8950 ± 11.7388	639.3058 ± 10.7563	79.7795 ± 39.9028
Walker Walk	703.8418 ± 27.3761	601.7267 ± 26.7561	95.9638 ± 21.2174
Cartpole Swingup	764.7615 ± 5.8841	710.6911 ± 31.6704	102.1589 ± 14.0952
Cup Catch	861.7760 ± 14.6734	789.6590 ± 27.4923	166.6440 ± 46.7917

Table C.7.: Mean and standard-error of final performance for the comparison of VRKN and RSSM on all tasks for the wall occlusion task.

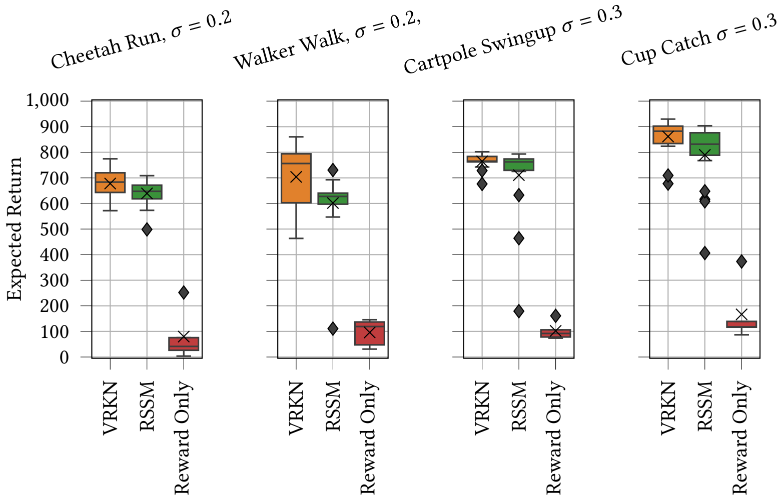


Figure C.13.: Box plots of final performance for the comparison of VRKN and RSSM on all tasks for the wall occlusion task.

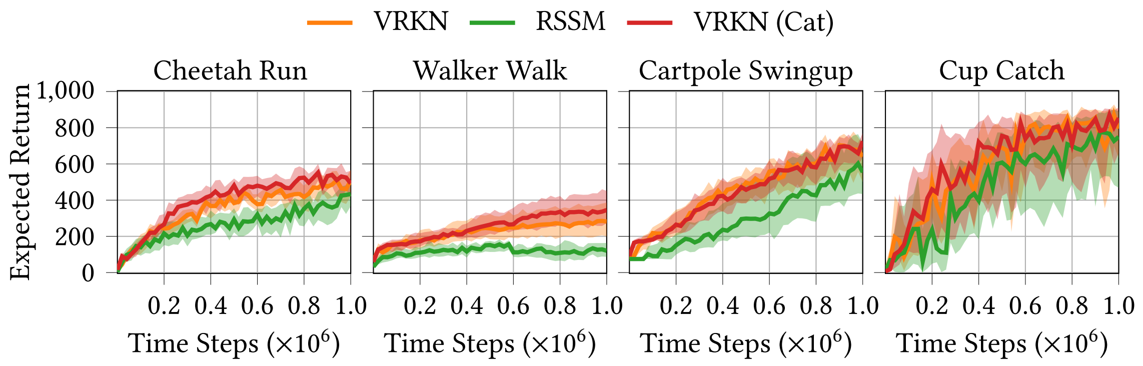


Figure C.14.: Comparison of VRKN and RSSM on all tasks considered with missing observations and without transition noise.

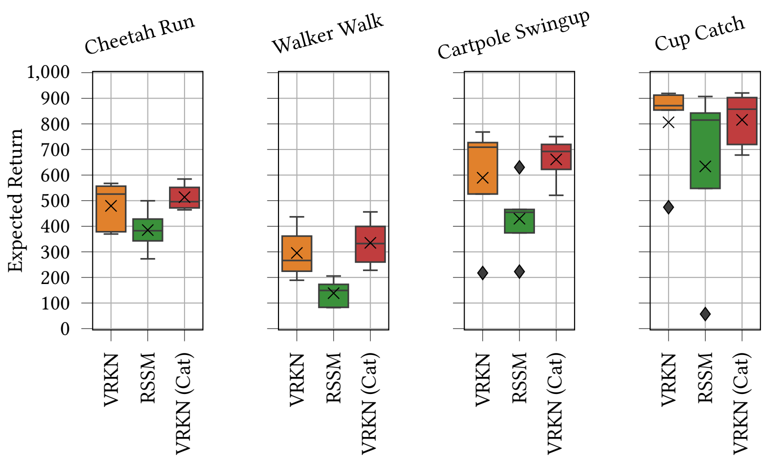


Figure C.15.: Box plots of final performance for the comparison of VRKN and RSSM on all tasks for the missing observation task without transition noise.

	VRKN	RSSM	VRKN (Cat)
Cheetah Run	479.6113 ± 39.0438	385.1623 ± 34.2866	513.6416 ± 20.9897
Walker Walk	295.5923 ± 40.7882	138.5695 ± 21.9475	335.1612 ± 37.8743
Cartpole Swingup	589.5229 ± 91.1411	429.4998 ± 59.3501	661.1643 ± 36.6275
Cup Catch	806.2720 ± 75.0126	633.8480 ± 140.1743	815.7960 ± 44.0564

Table C.8.: Mean and standard-error of final performance for the comparison of VRKN and RSSM on all tasks for the missing observation without transition noise.

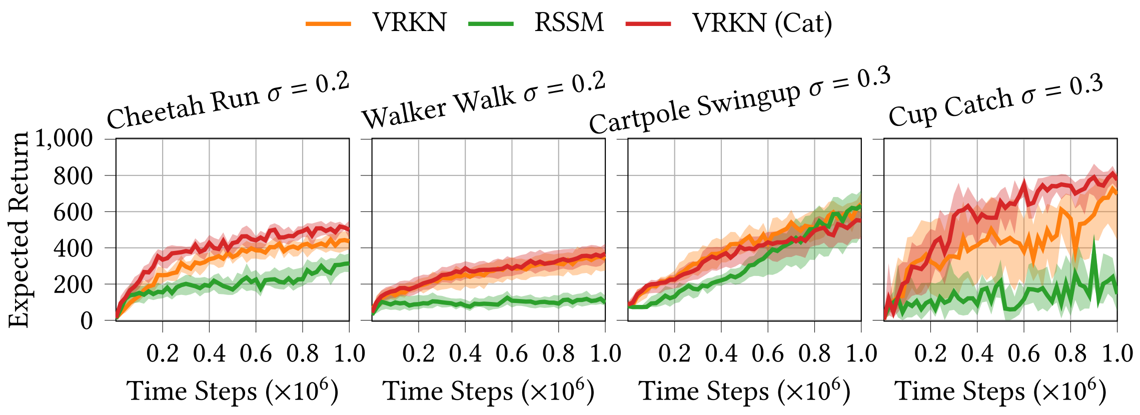


Figure C.16.: Comparison of VRKN and RSSM on all tasks considered with missing observations and transition noise. The σ in the title indicates the standard deviation of the Gaussian transition noise.

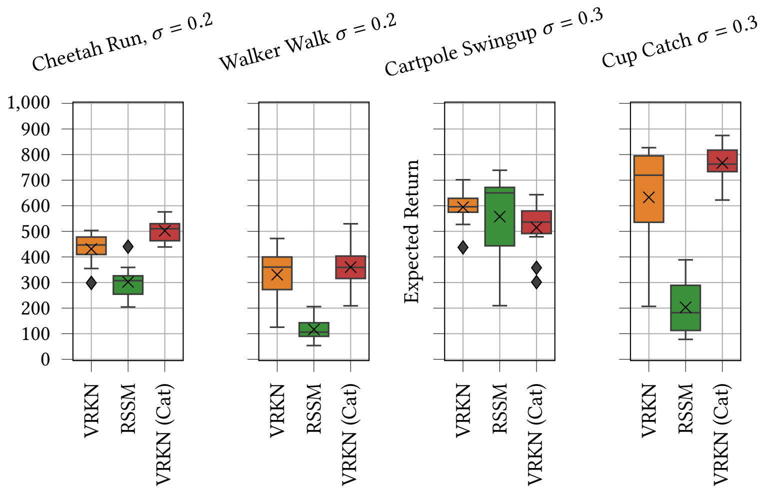


Figure C.17.: Box plots of final performance for the comparison of VRKN and RSSM on all tasks for the missing observation task with transition noise.

	VRKN	RSSM	VRKN (Cat)
Cheetah Run	431.1943 ± 19.2437	302.2760 ± 20.2040	502.3162 ± 13.2101
Walker Walk	330.7420 ± 30.5173	116.6937 ± 13.0846	360.0558 ± 26.1748
Cartpole Swingup	594.6512 ± 22.4822	557.7137 ± 58.3520	514.7862 ± 32.5809
Cup Catch	632.9800 ± 64.4935	203.1760 ± 32.3139	767.1400 ± 21.0592

Table C.9.: Mean and standard-error of final performance for the comparison of VRKN and RSSM on all tasks for the missing observation task with transition noise.

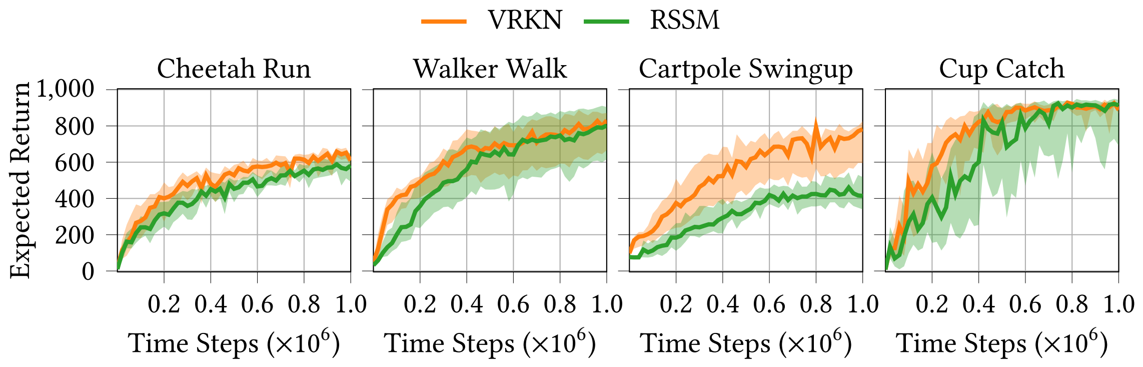


Figure C.18.: Comparison of VRKN and RSSM on all tasks considered for the sensor fusion task without transition noise.

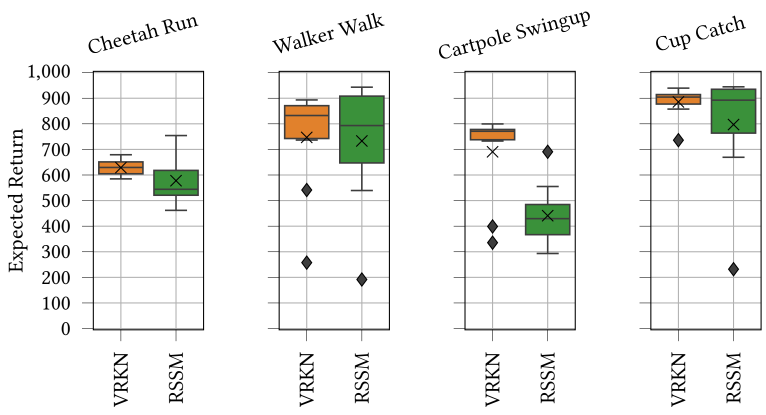


Figure C.19.: Box plots of final performance for the comparison of VRKN and RSSM on all tasks for the sensor fusion task without transition noise.

	VRKN	RSSM
Cheetah Run	628.8849 ± 9.6539	577.7296 ± 26.4294
Walker Walk	746.9262 ± 60.4730	733.0111 ± 70.6512
Cartpole Swingup	690.6291 ± 51.6357	440.9525 ± 36.0824
Cup Catch	885.5980 ± 17.4907	797.1000 ± 66.0047

Table C.10.: Mean and standard-error of final performance for the comparison of VRKN and RSSM on all tasks for the sensor fusion task without transition noise.

	VRKN	RSSM
Cheetah Run	594.5483 ± 13.6106	480.7173 ± 18.8966
Walker Walk	663.4760 ± 68.0791	635.1461 ± 68.1526
Cartpole Swingup	546.1525 ± 64.0839	354.1156 ± 51.2910
Cup Catch	809.5620 ± 69.0404	478.1520 ± 60.6500

Table C.11.: Mean and standard-error of final performance for the comparison of VRKN and RSSM on all tasks for the sensor fusion task with transition noise.

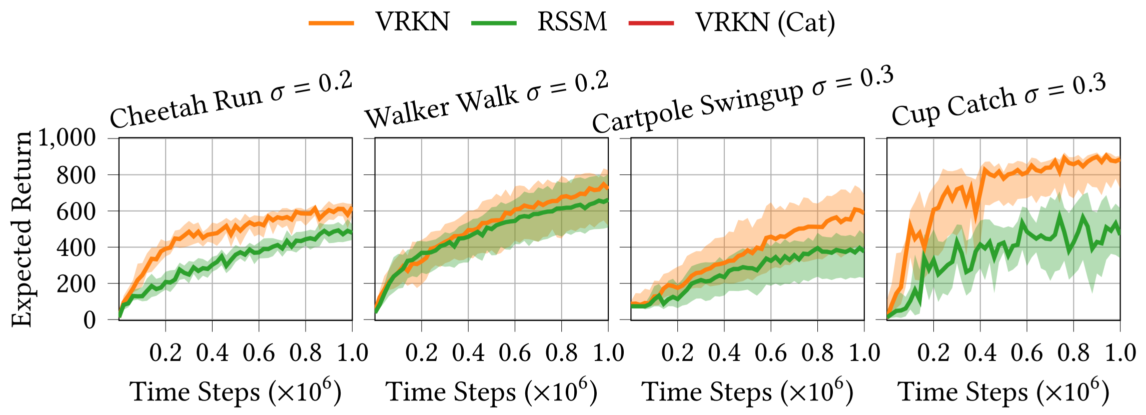


Figure C.20.: Comparison of VRKN and RSSM on all tasks considered for the sensor fusion task with transition noise. The σ in the title indicates the standard deviation of the Gaussian transition noise.

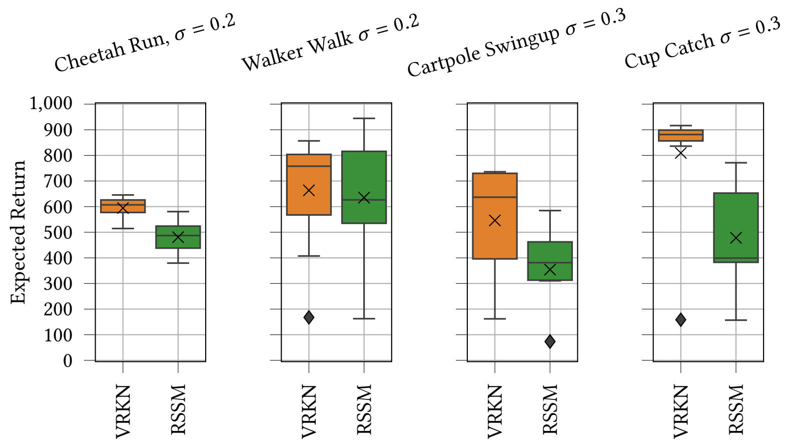


Figure C.21.: Box plots of final performance for the comparison of VRKN and RSSM on all tasks for the sensor fusion task with transition noise.

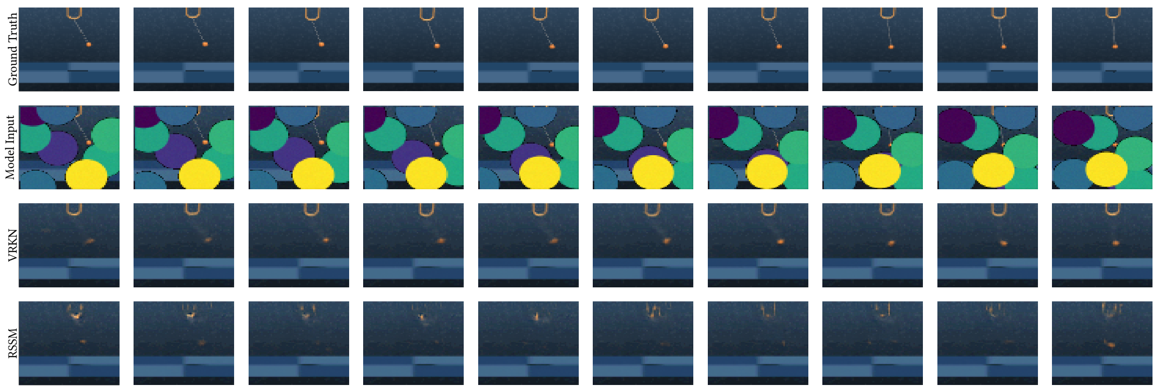


Figure C.22.: Trajectory samples of Cup Catch task with disc occlusions.

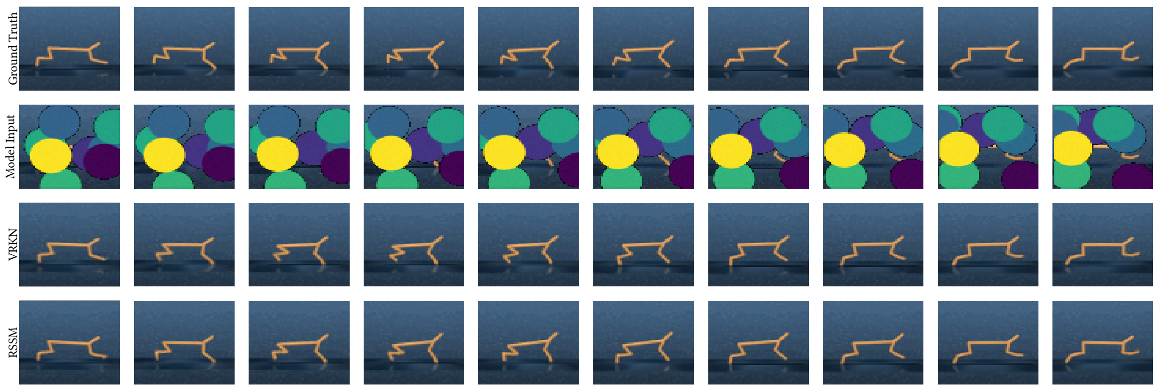


Figure C.23.: Trajectory samples of the Cheetah Run task with disc occlusions.

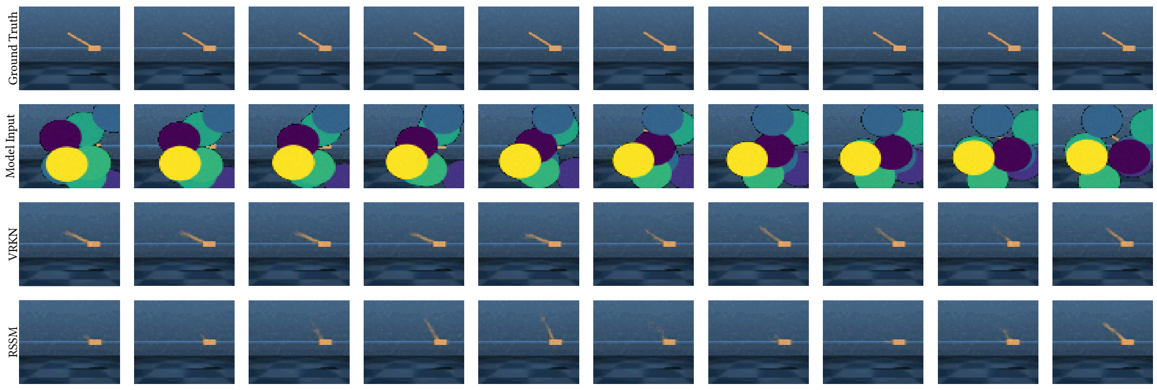


Figure C.24.: Trajectory samples of the Cartpole Swingup task with disc occlusions.

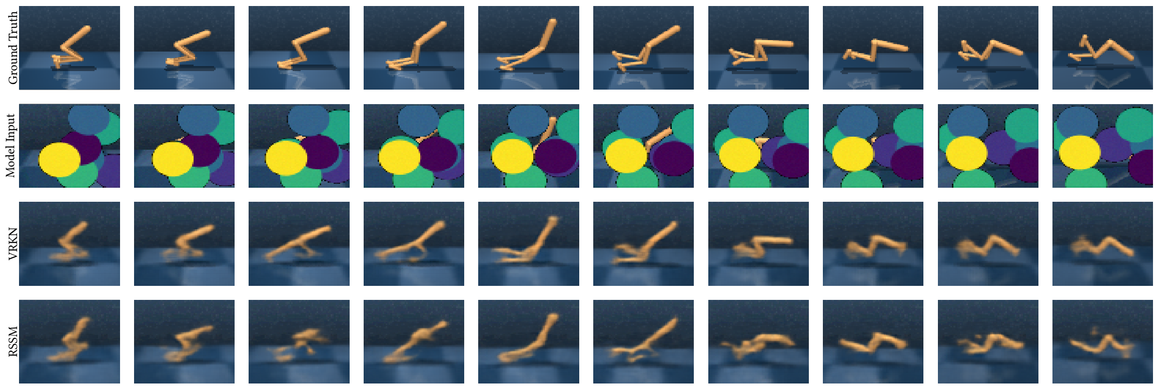


Figure C.25.: Trajectory samples of the Walker Walk task with disc occlusions.

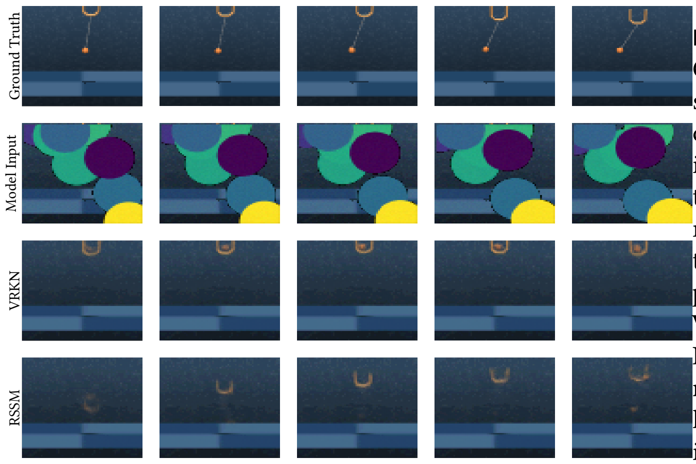


Figure C.26.: Another, sub-sequence of a Cup Catch trajectory. The first image shown here is the first image in the sequence. Neither the ball nor the cup is visible, i.e., the approaches have no information about their whereabouts and have to rely on their initial belief. As the model, towards the end of the agent's training, saw plenty of trajectories with caught balls, the VRKN's initial belief appears reasonable. Further, the VRKN's belief does only change minimally over time while the RSSM hallucinates movements without any visual information indicating them.

D. Appendix for Chapter 6

D.1. Hyperparameters and Implementation Details

Table D.1 lists all hyperparameters of the KalMamba model and Table D.2 lists the hyperparameters of Soft Actor Critic (SAC) (Haarnoja et al., 2018) used for control.

D.1.1. Baselines.

Both RSSM+SAC and VRKN+SAC use the same hyperparameters as KalMamba where applicable. For all other hyperparameters, we use the defaults from (Hafner et al., 2020) and (Becker and Neumann, 2022) respectively. The SAC baseline uses the hyperparameters listed in Table D.2 and the results for DreamerV3 (Hafner et al., 2023) are provided by the authors[1].

D.2. Additional Results

We provide results for the individual tasks of the Deepmind Control Suite for image-based observations in Figure D.1 and the different KalMamba ablations in Figure D.2. Figure D.3, shows the per-task results for the noisy state-based environments.

[1] `https://github.com/danijar/dreamerv3`

Hyperparameter	Low Dimensional DMC	Image Based DMC
	World Model	
Encoder	2×256 Unit NN with ELU	ConvNet from (Ha and Schmidhuber, 2018) and (Hafner et al., 2020) with ReLU
Decoder	2×256 Unit NN with ELU	ConvNet from (Ha and Schmidhuber, 2018) and (Hafner et al., 2020) with ReLU
Reward Decoder	2×256 Unit NN with ELU	
Latent Space Size	230 (30 Stoch. + 200 Det. for RSSM	
	Mamba Backbone	
num blocks	2	
d_model	256	
d_state	64	
d_conv	2	
dropout probability	0.1	
activation	SiLU	
pre mamba layers	2×256 Unit NN with SiLU	
post mamba layers	VRKN Dynamics Model Architecture from (Becker and Neumann, 2022) with SiLU	
	Loss	
KL Balancing	0.8 for RSSM, 0.5 for VRKN, KalMamba	
Free Nats	3	
α (regularization scale)	1, KalMamba only	
	Optimizer (Adam (Kingma and Ba, 2014))	
Learning Rate	$3 \cdot 10^{-4}$	

Table D.1.: World Model hyperparameters for KalMamba experiments

Hyperparameter	Low Dimensional DMC	Image Based DMC
Actor-Network	2×256 Unit NN with ReLU	3×1024 Unit NN with ELU
Critic-Network	2×256 Unit NN with ReLU	3×1024 Unit NN with ELU
Actor Optimizer	Adam with learning rate 3×10^{-4}	
Critic Optimizer	Adam with learning rate 3×10^{-4}	
Target Critic Update Fraction	0.005	
Target Critic Update Interval	1	
Target Entropy	$-d_{\text{action}}$	
Entropy Optimizer	Adam with learning rate 3×10^{-4}	
Initial Learning Rate	0.1	
discount γ	0.99	

Table D.2.: SAC hyperparameters for KalMamba experiments

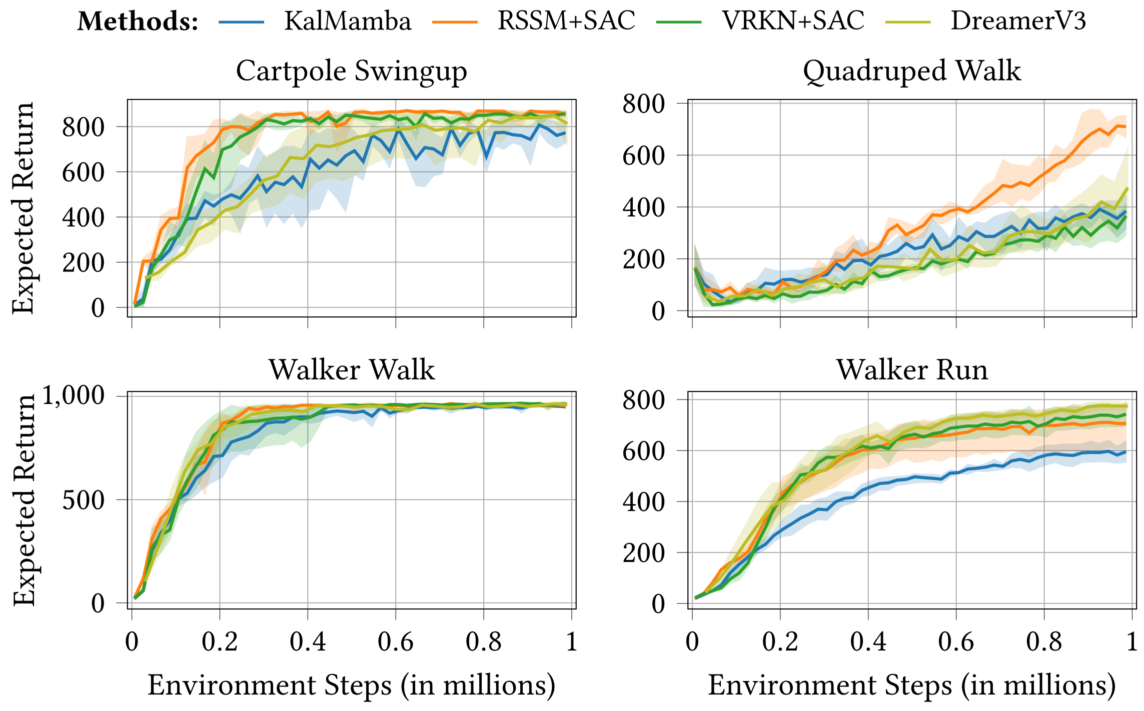


Figure D.1.: Per environment evaluations of the DeepMind Control Suite on image-based observations. Dreamer-v3 shows a performance similar to RSSM+SAC.

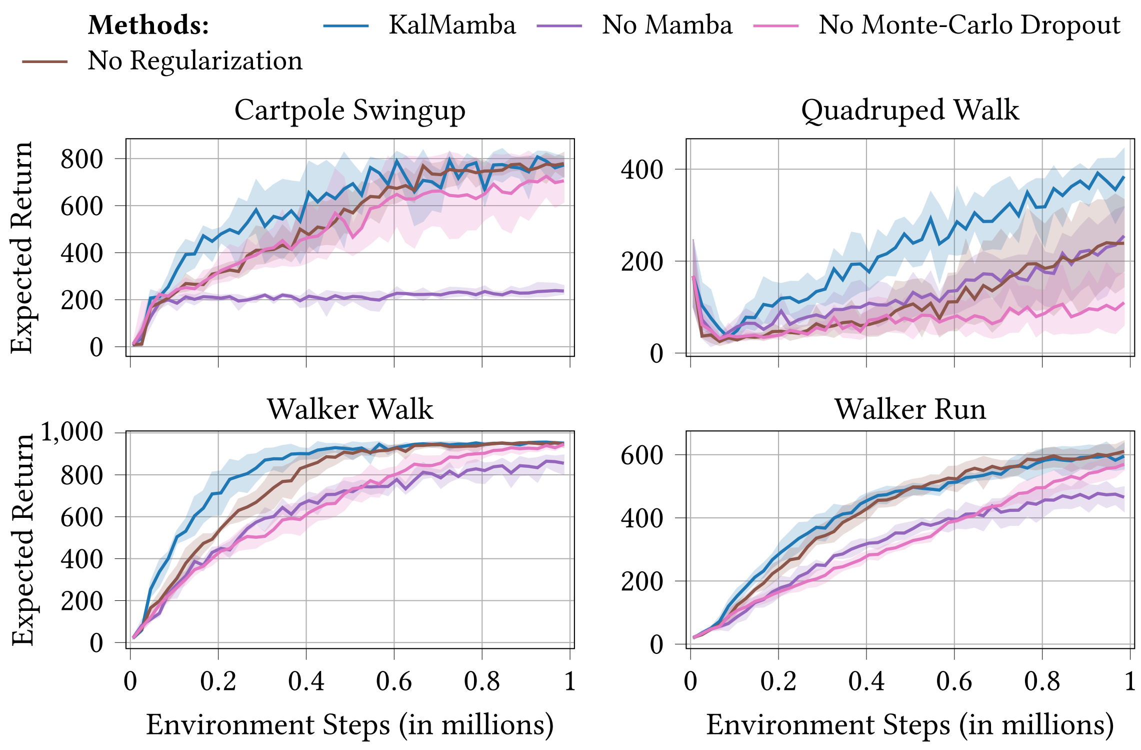


Figure D.2.: Per environment evaluations of the DeepMind Control Suite for different KalMamba ablations. Monte-Carlo Dropout and the Mahalanobis regularization make the largest difference for the hardest task in the suite, i.e., `quadruped_walk`.

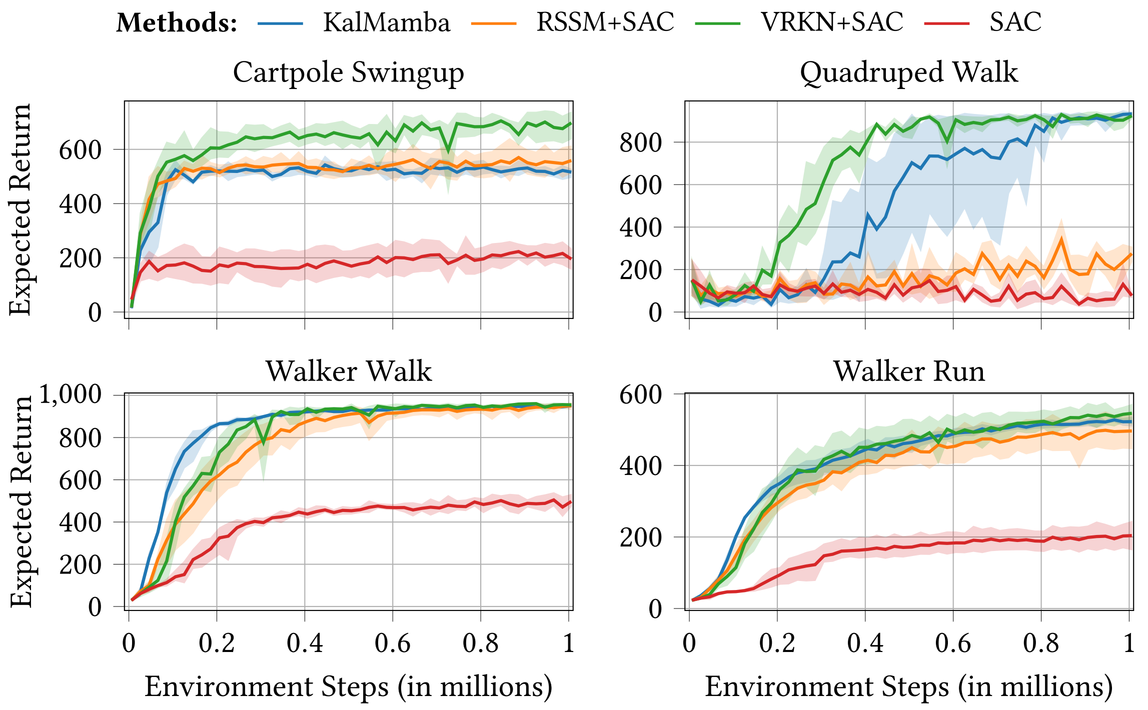


Figure D.3.: Per environment evaluations of the DeepMind Control Suite on low-dimensional state representations. KalMamba performs on par with or better than the RSSM on all tasks, and is only outperformed by the computationally more expensive VRKN on `cartpole_ swingup`.

E. Appendix for Chapter 7

E.1. Architecture Details and Training

We use the same hyperparameters for all experiments based on the DeepMind Control Suite (DMC), i.e., the standard tasks with the different observation types (Video Background, Occlusions and also Standard Images) as well as, the Locomotion Suite. For the ManiSkill2-based Manipulation Suite, we use a larger model and a more conservative update scheme for actors and critics. We use the ELU activation function unless otherwise mentioned.

E.1.1. Environments

For the model-based agents, we followed common practice (Hafner et al., 2020; Fu et al., 2021; Wang et al., 2022; Deng et al., 2022) and use an action repeat of 2 for all environments. We do the same for the model-free agents except for: `Ball In Cup Catch` (4), `Cartpole Swingup` (8), `Cheetah Run` (4) and `Reacher Easy` (4). All environments in the locomotion suite also use an action repeat of 2, this includes `Hurdle Cheetah Run` which requires more fine-grained control than the normal version to avoid the hurdles.

The environments of the Manipulation suite do not use action repeat. Here, we use delta joint position control, no action repeat, and dense normalized rewards for all tasks. For the depth images we use the depth camera functionality provided by ManiSkill2 and clip to values between 0 and 4 meters.

E.1.2. Recurrent State Space Model

We denote the deterministic part of the RSSM's state by $\mathbf{h}_t$ and the stochastic part by $\mathbf{s}_t$. The base RSSM model without parts specific to the objective consists of:

- **Encoders:** $\psi_{\text{obs}}^{(k)}(\mathbf{o}_t)$, where ψ_{obs} is the convolutional architecture proposed by (Ha and Schmidhuber, 2018) and used by (Hafner et al., 2019, 2020) for image observations. For the low-dimensional proprioception, we used 3×400 fully connected layers for the DMC tasks and 4×512 fully connected layers Manipulation Suite.
- **Deterministic Path**: $\mathbf{h}_t = g(\mathbf{z}_{t-1}, \mathbf{a}_{t-1}, \mathbf{h}_{t-1}) = \text{GRU}(\psi_{\text{det}}(\mathbf{z}_{t-1}, \mathbf{a}_{t-1}), \mathbf{h}_{t-1})$ (Cho et al., 2014), where ψ_{det} is a 2×400 fully connected NN and the GRU has a memory size of 200 for the DMC tasks. For the Manipulation Suite ψ_{det} has 2×512 units and the GRU a memory size of 400

- **Dynamics Model**: $p(\mathbf{z}_{t+1}|\mathbf{z}_t, \mathbf{a}_t) = \psi_{\text{dyn}}(\mathbf{h}_t)$, where ψ_{dyn} is a 2×400 units fully connected NN for the DMC tasks and a 2×512 units fully connected NN for the Manipulation Suite. The network learns the mean and standard deviation of the distribution.
- **Variational Distribution** $q(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{a}_{t-1}, \mathbf{o}_t) = \psi_{\text{var}}\left(\mathbf{h}_t, \text{Concat}\left(\{\psi_{\text{obs}}^{(k)}(\mathbf{o}_t^{(k)})\}_{k=1:K}\right)\right)$, where ψ_{var} is a 2×400 units fully connected NN for the DMC tasks and a 2×512 units fully connected NN for the Manipulation Suite. Again, the network learns the mean and standard deviation of the distribution.
- **Reward Predictor** $p(r_t|\mathbf{z}_t)$: 2×128 units fully connected NN for model-free agents. 3×300 units fully connected NN with ELU activation for model-based agents. The network only learns the mean of the distribution. The standard deviation is fixed at 1. The model-based agents use a larger reward predictor as they rely on it for learning the policy and the value function. Model-free agents use the reward predictor only for representation learning and work with the ground truth rewards from the replay buffer to learn the critic.

E.1.3. Objectives

Image Inputs and Augmentation. Whenever we use a contrastive image loss, we randomly crop a 64×64 pixel image from the original image of size 76×76 pixels during training. Cropping is temporally consistent, i.e., the same crop is used for all time steps in a sub-sequence. For evaluation, we corp at the center. For the ablations that reconstruct images, we downsize them directly to 64×64 pixels.

KL. For the KL terms in Equation 7.1 and Equation 7.3 we follow Hafner et al. (2023) and combine the KL-Balancing technique introduced by Hafner et al. (2021) with the free-nats regularization used by Hafner et al. (2019, 2020). Following Hafner et al. (2021) we use a balancing factor of 0.8. We give the algorithm 1 free nat for the DMC Tasks and 3 for the Manipulation Suite.

Following Srivastava et al. (2021) we scale the KL term using a factor of $\beta = 0.001$ for the contrastive predictive objective.

Reconstruction Objectives. Whenever we reconstruct images we use the up-convolutional architecture proposed by (Ha and Schmidhuber, 2018) and used by (Hafner et al., 2019, 2020). For low-dimensional observations, we use 3×400 units fully connected NN for the DMC tasks and a 4×512 Units fully connected NN for the Manipulation Suite. In all cases, only the mean is learned. We use a fixed variance of 1 for all image losses and the proprioception for the DMC tasks. For the Manipulation Suite, we set the variance for the proprioception to 0.04.

Optimizer. We used Adam (Kingma and Ba, 2014) with $\alpha = 3 \times 10^{-4}$, $\beta_1 = 0.99, \beta_2 = 0.9$ and $\varepsilon = 10^{-8}$ for all losses. We clip gradients if the norm exceeds 10.

E.1.4. Soft Actor Critic

Table E.1 lists the hyperparameters used for model-free RL with SAC (Haarnoja et al., 2018).

We collected $5,000$ initial steps at random. During training, we update the RSSM, critic, and actor in an alternating fashion for d steps before collecting a new sequence by directly sampling from

Hyperparameter	DMC and Locomotion	Manipulation
Actor Hidden Layers	$3 \times 1,024$ Units	$3 \times 1,024$ Units
Actor Activation	ELU	ELU + LayerNorm
Critic Hidden Layers	$3 \times 1,024$ Units	$3 \times 1,024$ Units
Critic Activation	Tanh	ELU + LayerNorm
Discount	0.99	0.85
Actor Learning Rate	0.001	0.0003
Actor Gradient Clip Norm	10	10
Critic Learning Rate	0.001	0.0003
Critic Gradient Clip Norm	100	100
Target Critic Decay	0.995	0.995
Target Critic Update Interval	1	1
α learning rate	0.001	0.0003
initial α	0.1	1.0
target entropy	- action dim	- action dim

Table E.1.: Hyperparameters used for policy learning with the Soft Actor-Critic.

the maximum entropy policy. Here, d is set to be half of the environment steps collected per sequence (after accounting for potential action repeats). Each step uses 32 subsequences of length 32, uniformly sampled from all prior experience.

E.1.5. Latent Imagination

Table E.2 lists the hyperparameters used for model-based RL with latent imagination. They follow to a large extent those used in (Hafner et al., 2020, 2021).

We again collect $5,000$ initial steps at random. During training, we update the RSSM, value function, and actor in an alternating fashion for 100 steps before collecting new sequences. Each step uses 50 subsequences of length 50, uniformly sampled from all prior experience. For collecting new data, we use constant Gaussian exploration noise with $\sigma = 0.3$.

E.2. Details on Baselines and Ablations.

For Dreamer-v3 (Hafner et al., 2023) we use the raw reward curve data provided with the official implementation[1]. For DreamerPro (Deng et al., 2022)[2], Task Informed Abstractions (Fu et al.,

[1] `https://github.com/danijar/dreamerv3/blob/main/scores/data/dmcvision_dreamerv3.json.gz`
[2] `https://github.com/fdeng18/dreamer-pro`

Hyperparameter	Value
Actor Hidden Layers	3×300 Units
Actor Activation	ELU
Critic Hidden Layers	3×300 Units
Critic Activation	ELU
Discount	0.99
Actor Learning Rate	8×10^{-5}
Actor Gradient Clip Norm	100
Value Function Learning Rate	8×10^{-5}
Value Gradient Clip Norm	100
Slow Value Decay	0.98
Slow Value Update Interval	1
Slow Value Regularizer	1
Imagination Horizon	15
Return lambda	0.95

Table E.2.: Hyperparameters used for policy learning with Latent Imagination.

2021)[3], Deep Bisumlation for Control (Zhang et al., 2020)[4], DenoisedMDP (Wang et al., 2022)[5] and DrQ-v2 (Yarats et al., 2022)[6] we use the official implementations provided by the respective authors.

DrQ-(I+P) builds on the official implementation and uses a separate encoder for the proprioception whose output is concatenated to the image encoder's output and trained using the critics' gradients.

We implemented RePo and RePo(I+P) in our framework, reused the Hyperparameters form Zhu et al. (2023), and ensured the results of our implementation match the official implementation's[7] result on the DMC tasks with standard images. RePo(I+P) encodes the proprioception using a separate encoder and both the embedded image and proprioception are given to the RSSM.

E.2.1. Hyperparameters of Abltions and Baselines.

Ablations. All Same-Loss, Concat, and Img-Only use the hyperparameters listed in Appendix E.1. They are merely missing certain parts of the model or use a different loss for one or both modalities. For the Concat baseline, we project the proprioception to the RSSMs latent state size (stochastic + deterministic) using a single linear layer before concatenation.

[3] `https://github.com/kyonofx/tia/`
[4] `https://github.com/facebookresearch/deep_bisim4control/`
[5] `https://github.com/facebookresearch/denoised_mdp`
[6] `https://github.com/facebookresearch/drqv2`
[7] `https://github.com/zchuning/repo`

ProprioSAC uses the hyperparameters listed in Table E.1, except for the learning rates. We reduced those to the SAC default values of 0.0003 for all environments, as we found the more aggressive updates used for CoRAL on Video Background, Occlusions and Locomotion can lead to instabilities when training directly on the proprioception.

Baselines. All our baselines were originally evaluated on standard DeepMind Control Suite tasks, modified DeepMind Control Suite tasks, or both. They were designed for problems very similar to Occlusions and, in particular, Video Background and we thus reuse the Hyperparameters originally proposed by the respective authors. For baselines using an RSSM, (TIA, DreamerPro, DenoisedMDP, and RePo) these are generally very similar and follow (Hafner et al., 2020, 2021).

For the Locomotion suite all approaches, including CoRAL and the ablations, use the same Hyperparameters as they use for Video Backgrounds and Occlusions.

For the Manipiluation suite we increased the model sizes of RePo following those of CoRAL. For both the DrQ-v2-based and the RePo-based baselines we tried a discount factor of 0.85 and 0.99 to ensure the performance differences to CoRAL is not an artifact of the low discount of 0.85. However, the lower discount worked better for all methods.

E.2.2. On the Performance of Some Baselines in our Setting.

As described in Section 7.4, there are distinct ways how to select and use the Kinetics400 videos in the existing literature. Nguyen et al. (2021), who first introduced the more challenging setting we use, already found DBC (Zhang et al., 2020) to struggle in this setting and our results align with those findings.

TIA (Fu et al., 2021) and DenoisedMDP (Wang et al., 2022) factorize the latent variable into 2 distinct parts and formulate loss functions that force one part to focus on task-relevant aspects and the other on task-irrelevant aspects. However, the part responsible for the task-irrelevant aspects still has to model those explicitly. In the more complicated setting with randomly sampled, colored background videos, the TIA and DenoisedMDP world models underfit and thus fail to learn a good representation or policy. Contrastive approaches, such as our approach and DreamerPro (Deng et al., 2022), do not struggle with this issue, as they do not have to model task-irrelevant aspects but can learn to ignore them.

RePo (Zhu et al., 2023) was also evaluated on the simpler setting and Zhu et al. (2023) report an improved performance over TIA and DenoisedMDP. In the more challenging setting, this improvement persists and RePo performs similarly to DreamerPro (Figure 7.5).

Furthermore, Zhu et al. (2023) presents results on ManiSkill2 environments similar to `LiftCube`, `PushCube`, and `TurnFaucet` of our Manipulation Suite. However, as detailed in Section 7.4 our Manipulation Suite tasks randomize initial conditions (i.e., cube position or faucet model) which results in significantly more challenging tasks, in which RePo seems to struggle.

E.3. Complete Results

The following pages list the aggregated results and performance profiles for all tasks, representation-learning approaches, and both model-free and model-based RL. We compute inter-quartile means and stratified bootstrapped confidence intervals, as well as the performance profiles according to the recommendations of Agarwal et al. (2021) using the provided library[8]. For each task in the suites, we ran 5 seeds per method, i.e., the results for Standard Images, Video Backgrounds, and Occlusions are aggregated over 35 runs, and those for Locomotion and Manipulation over 30 runs. Figure E.1 lists the aggregated results for all model-free agents on the DeepMind Control (DMC) Suite tasks and Figure E.3 lists the corresponding performance profiles. Figure E.2 lists the aggregated results for all model-based agents on the DeepMind Control Suite tasks and Figure E.4 lists the corresponding performance profiles. Figure E.5 shows aggregated results and performance profiles for the Locomotion suite. Figure E.6 shows aggregated results and performance profiles for the Manipulation suite. We also list the per-task results for all task suits:

- Figure E.7: Model-free agents on DMC tasks with Standard Images
- Figure E.8: Model-free agents on DMC tasks with Video Background.
- Figure E.9: Model-free agents on DMC tasks with Occlusions.
- Figure E.10: Model-based agents on DMC tasks with Standard Images.
- Figure E.11: Model-based agents on DMC tasks with Video Background.
- Figure E.12: Model-based agents on DMC tasks with Occlusions.
- Figure E.13: Per Environment Results for the Locomotion suite.
- Figure E.14: Per Environment Results for the Manipulation suite.

[8] `https://github.com/google-research/rliable`

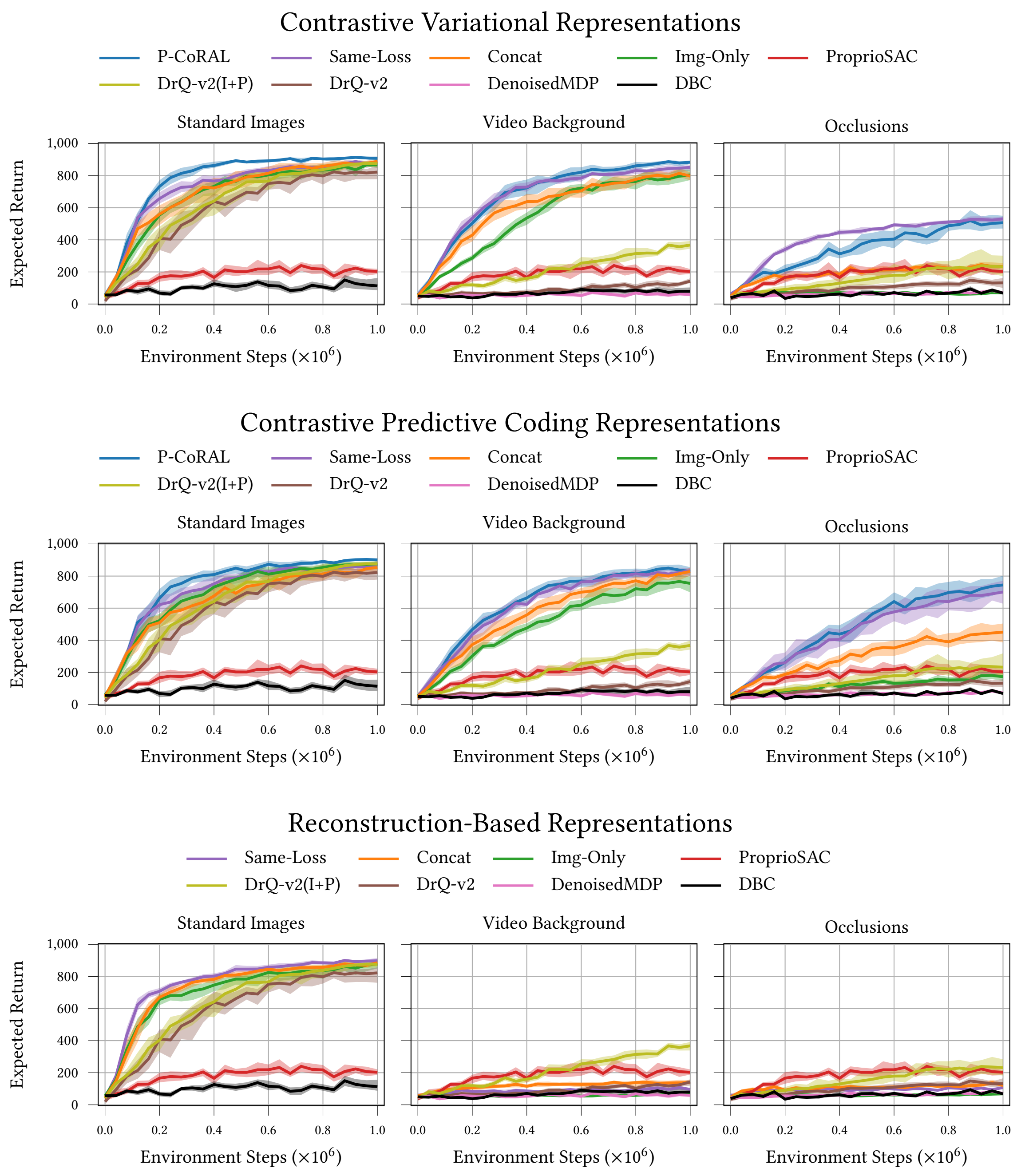


Figure E.1.: Aggregated results for all **model-free** agents on the DeepMind Control Suite environments with Standard Images, Video Background, and Occlusions. As expected, reconstruction-based approaches do not work on Video Background and Occlusions. Out of all considered approaches V-CoRAL achieves the highest performance on Video Background and P-CoRAL achieves the highest performance on Occlusions.

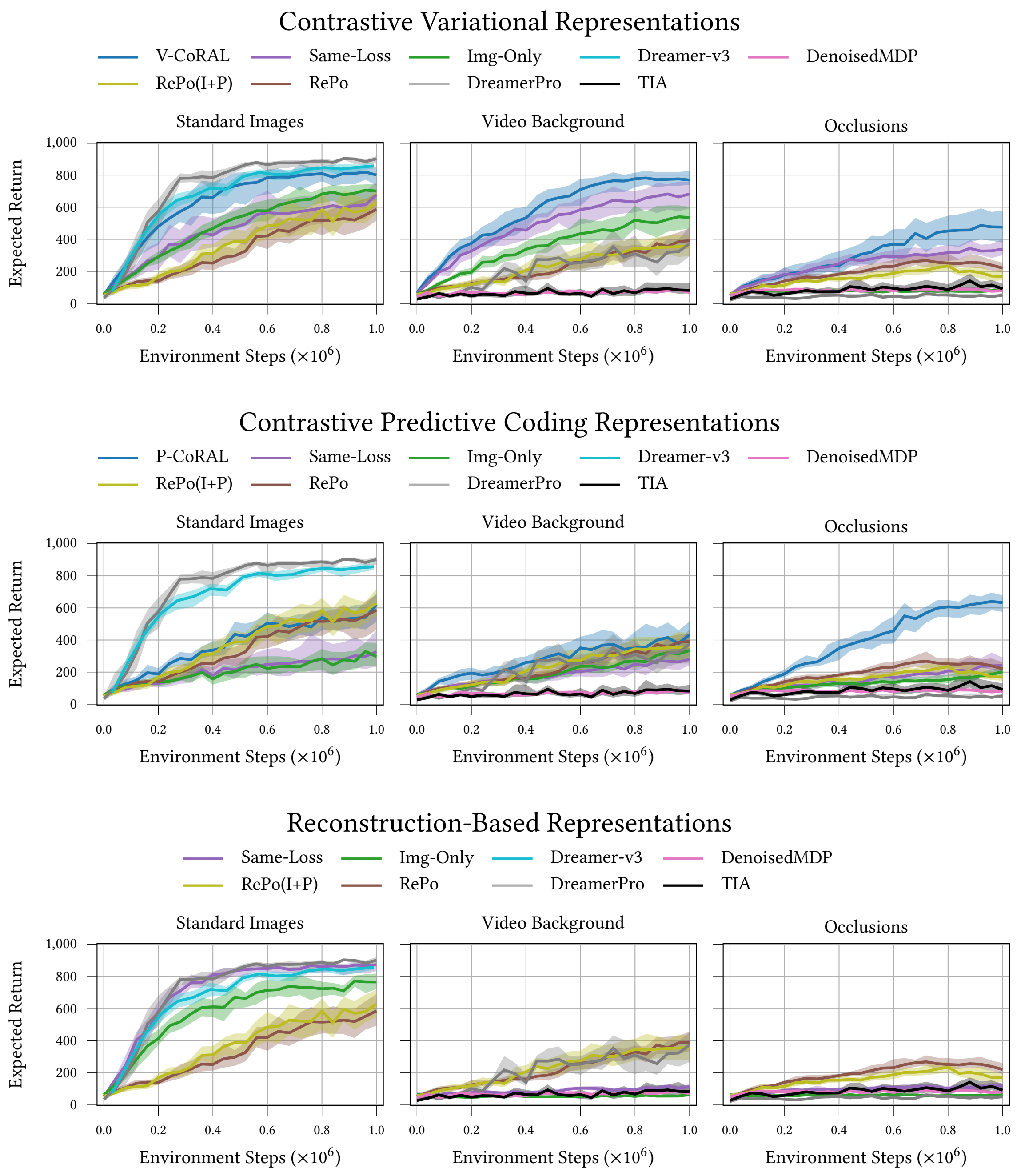


Figure E.2.: Aggregated results for all **model-based** agents on the DeepMind Control Suite environments with Standard Images, Video Background, and Occlusions. Compared to their model-free counterparts (Figure E.1), model-based agents perform worse, except if a reconstruction-based representation is used. Yet, the performance gap is larger for image-only and fully contrastive approaches. Especially V-CoRAL still achieves high performance on Video Background, almost matching the performance of Dreamer-v3 on Standard Images. This further highlights the benefits of using CoRAL, which can significantly improve over tailored approaches such as DreamerPro or RePo.

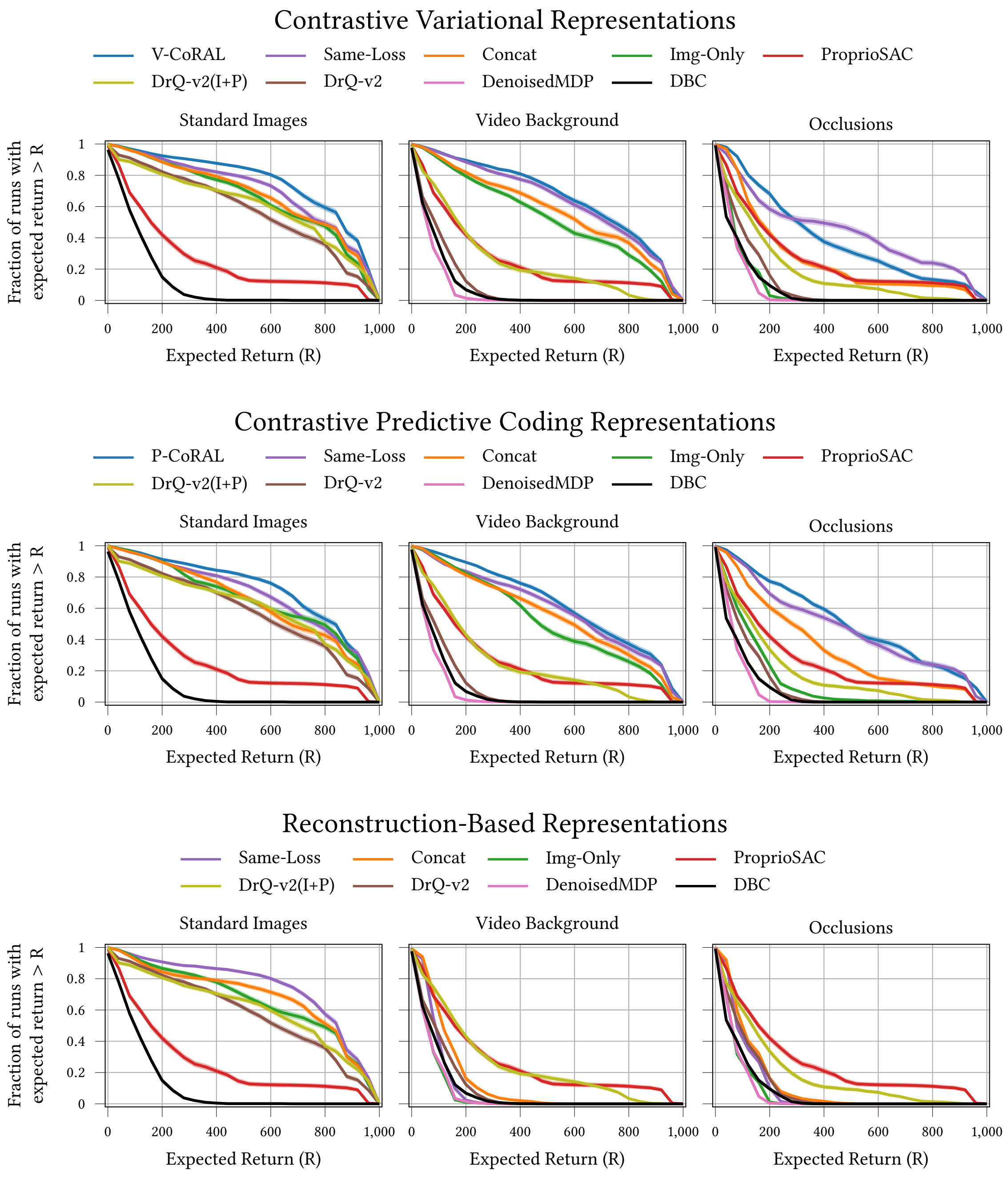


Figure E.3.: Performance profiles for all **model-free** agents on the DeepMind Control Suite tasks with Standard Images, Video Background, and Occlusions. They show that performance is largely consistent across the tasks. The sole exception is V-CoRAL and the contrastive variational approach with the same loss for both modalities on Occlusions. Here, the former fails for `Ball-in-Cup Catch` and `Cartpole Swingup`, while the latter underperforms for `Cheetah Run` (Figure E.9).

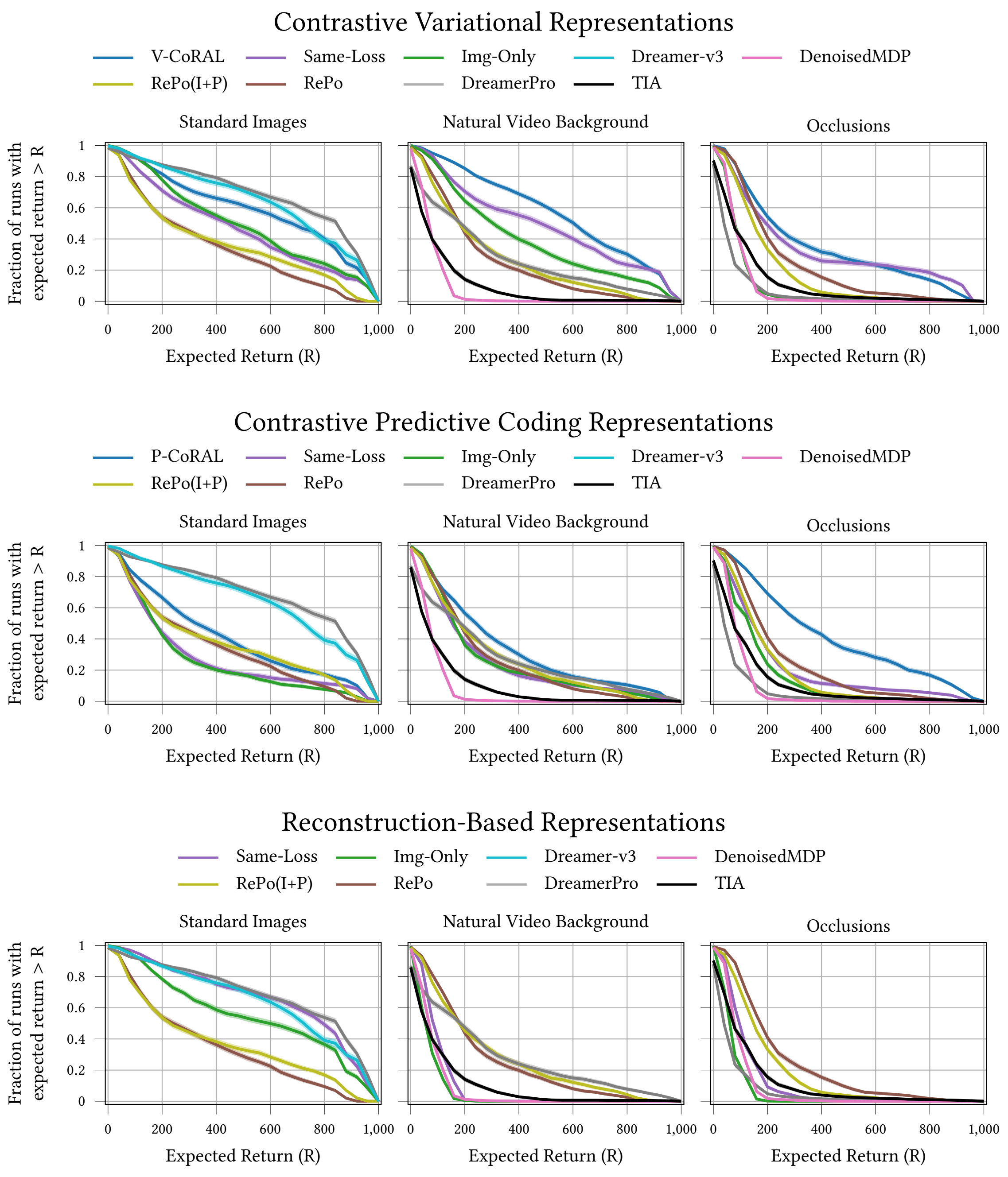


Figure E.4.: Performance profiles for all **model-based** agents on the DeepMind Control Suite environments with Standard Images, Video Background, and Occlusions. They indicate that performance is largely consistent across the environments.

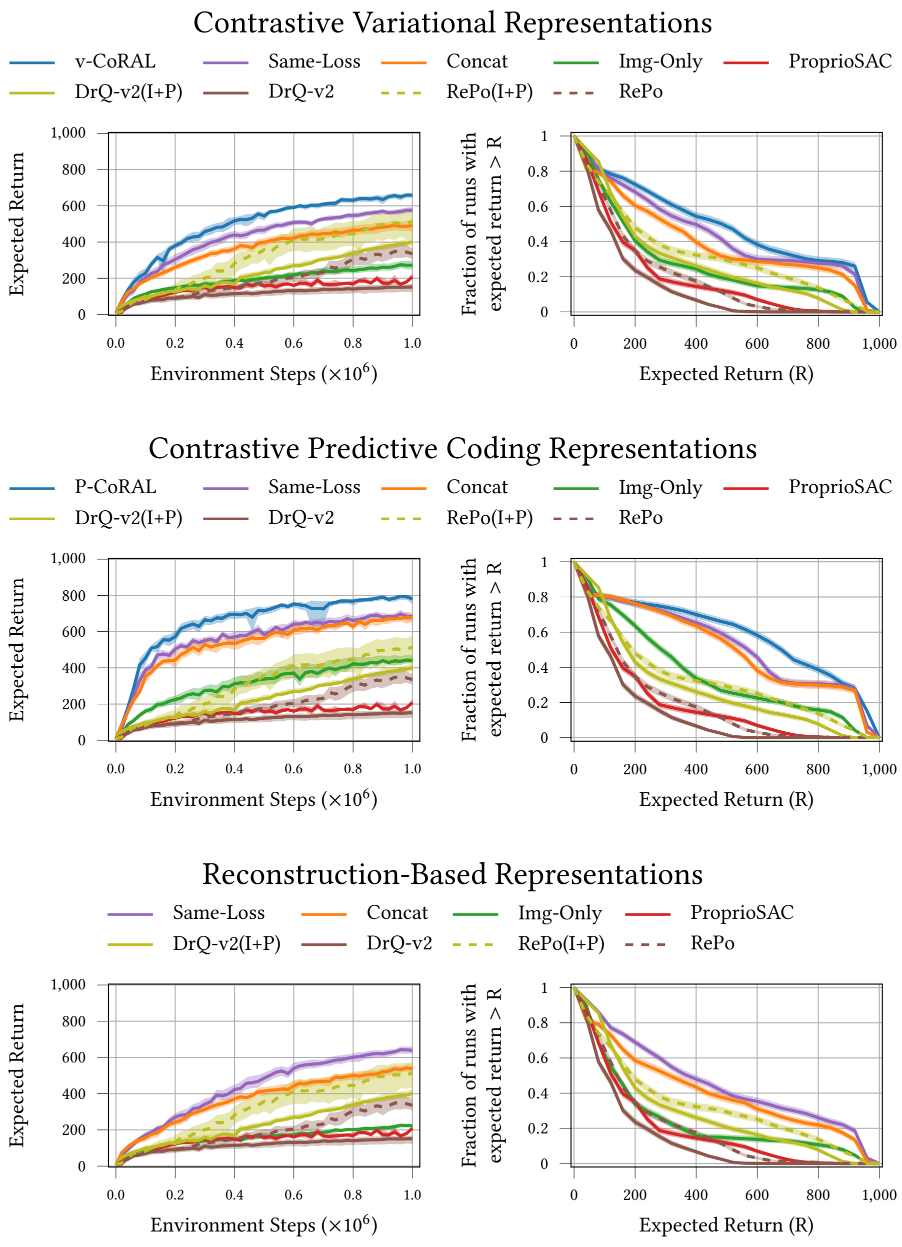


Figure E.5.: Aggregated results and performance profiles for the Locomotion suite. Both V-CoRAL and P-CoRAL outperform reconstruction and P-CoRAL gives the best results of all approaches by a significant margin Figure E.13 shows that the performance difference is larger in environments with randomly colored obstacles (`Hurdle Cheetah Run`, `Hurdle Walker Walk`, `Hurdle Walker Run`. The color is not relevant to avoid the obstacles but seems to hinder reconstruction.

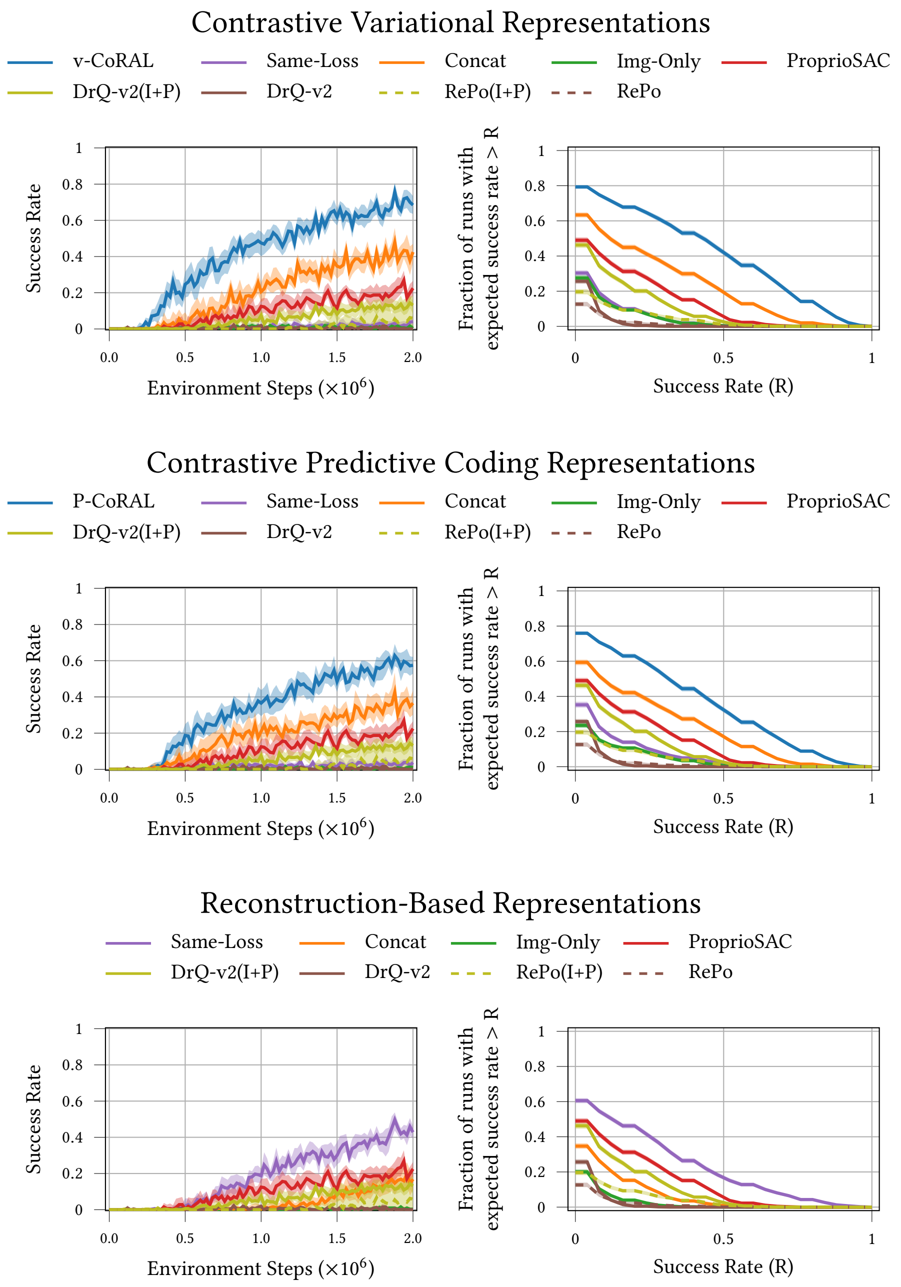


Figure E.6.: Aggregated results and performance profiles for the Manipulation Suite. V-CoRAL performs best by a significant margin, followed by P-CoRAL No approach that uses solely images, i.e., Img-Only-ablations, RePo and DrQ-v2, or uses both modalities but has a fully constrastive objective achieves any notable success.

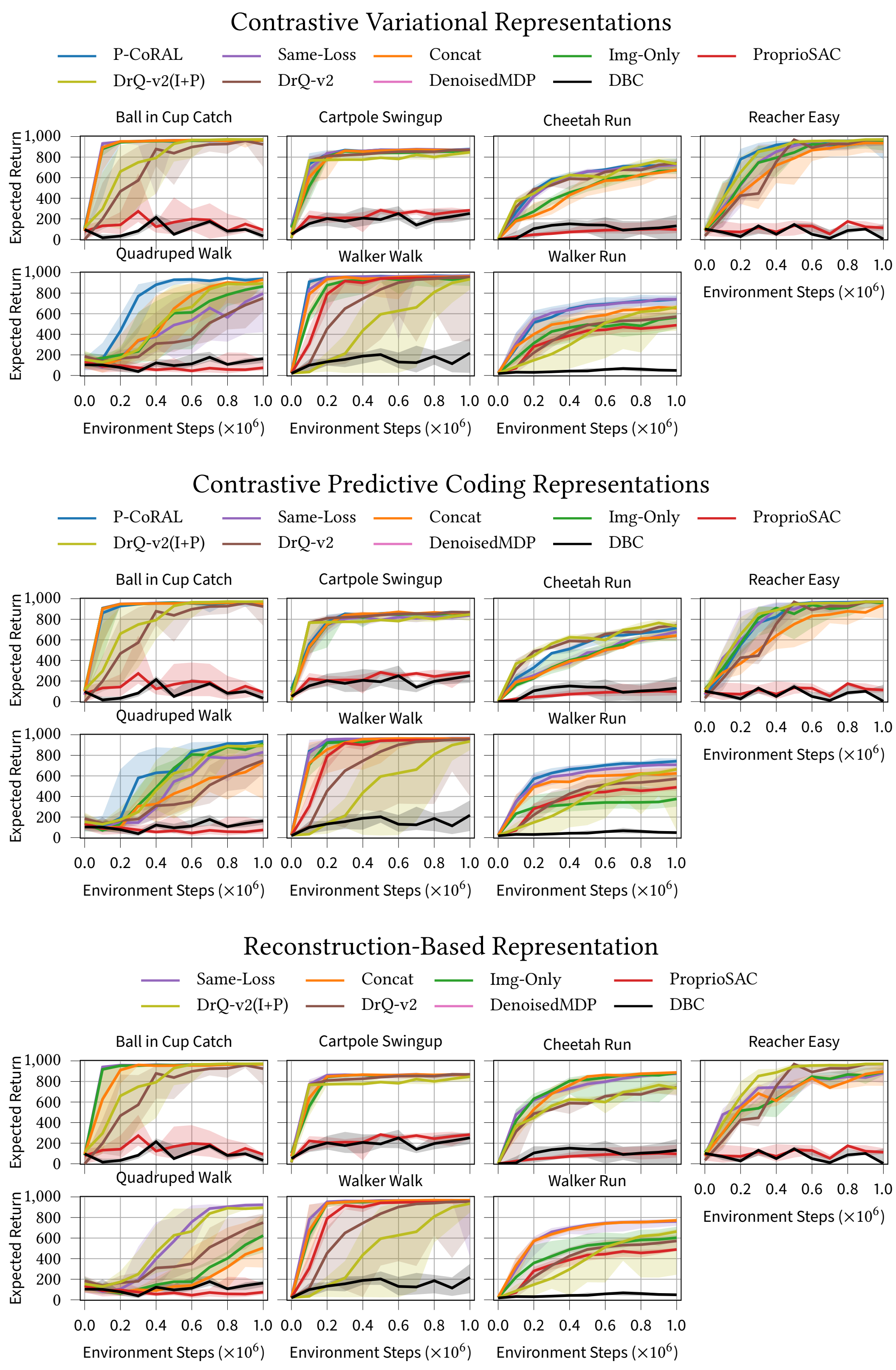


Figure E.7.: Per environment results for model-free agents on the DeepMind Control Suite with Standard Images.

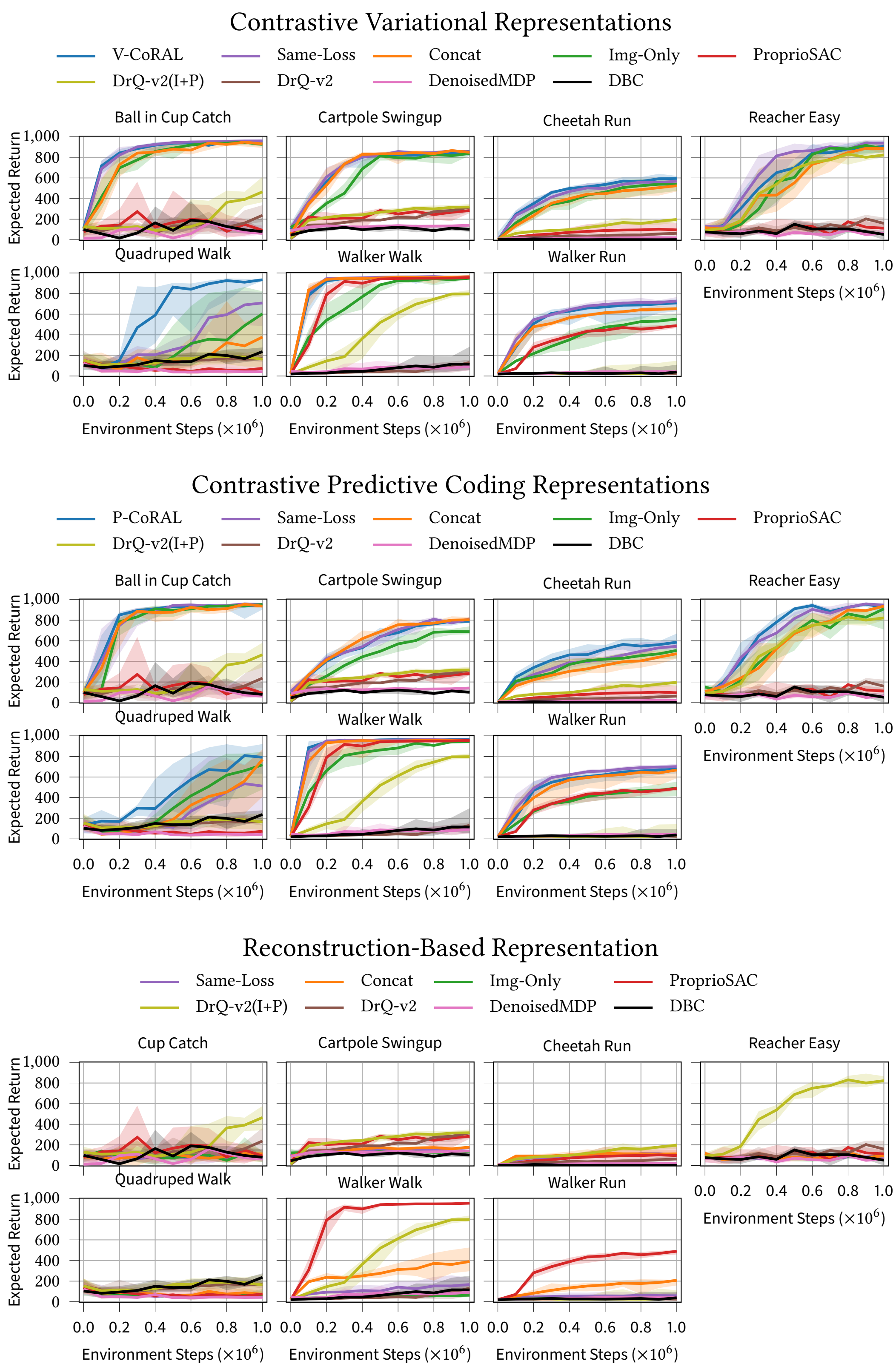


Figure E.8.: Per environment results for model-free agents on the DeepMind Control Suite with Video Background.

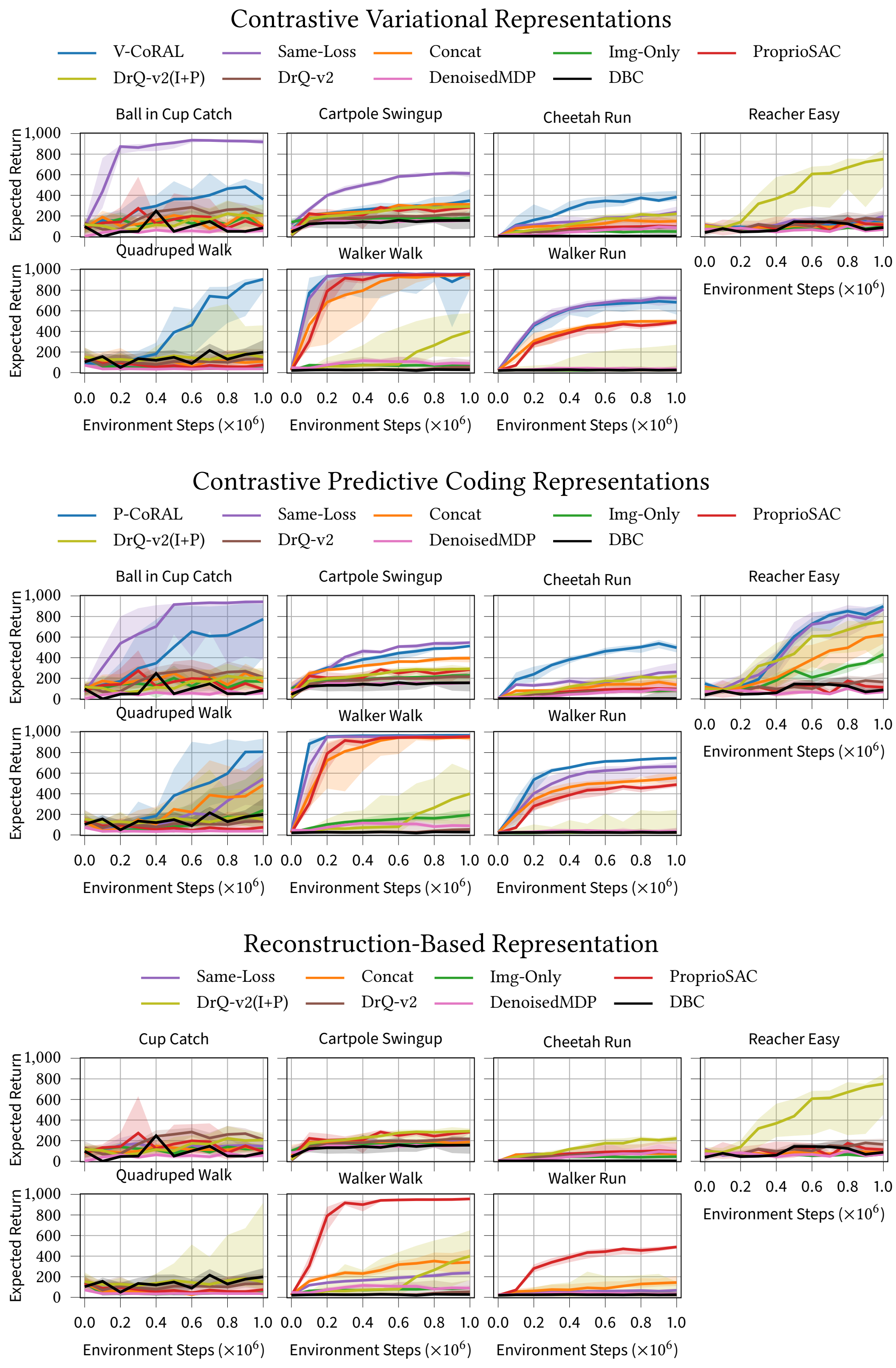


Figure E.9.: Per environment results for model-free agents on the DeepMind Control Suite with Occlusions.

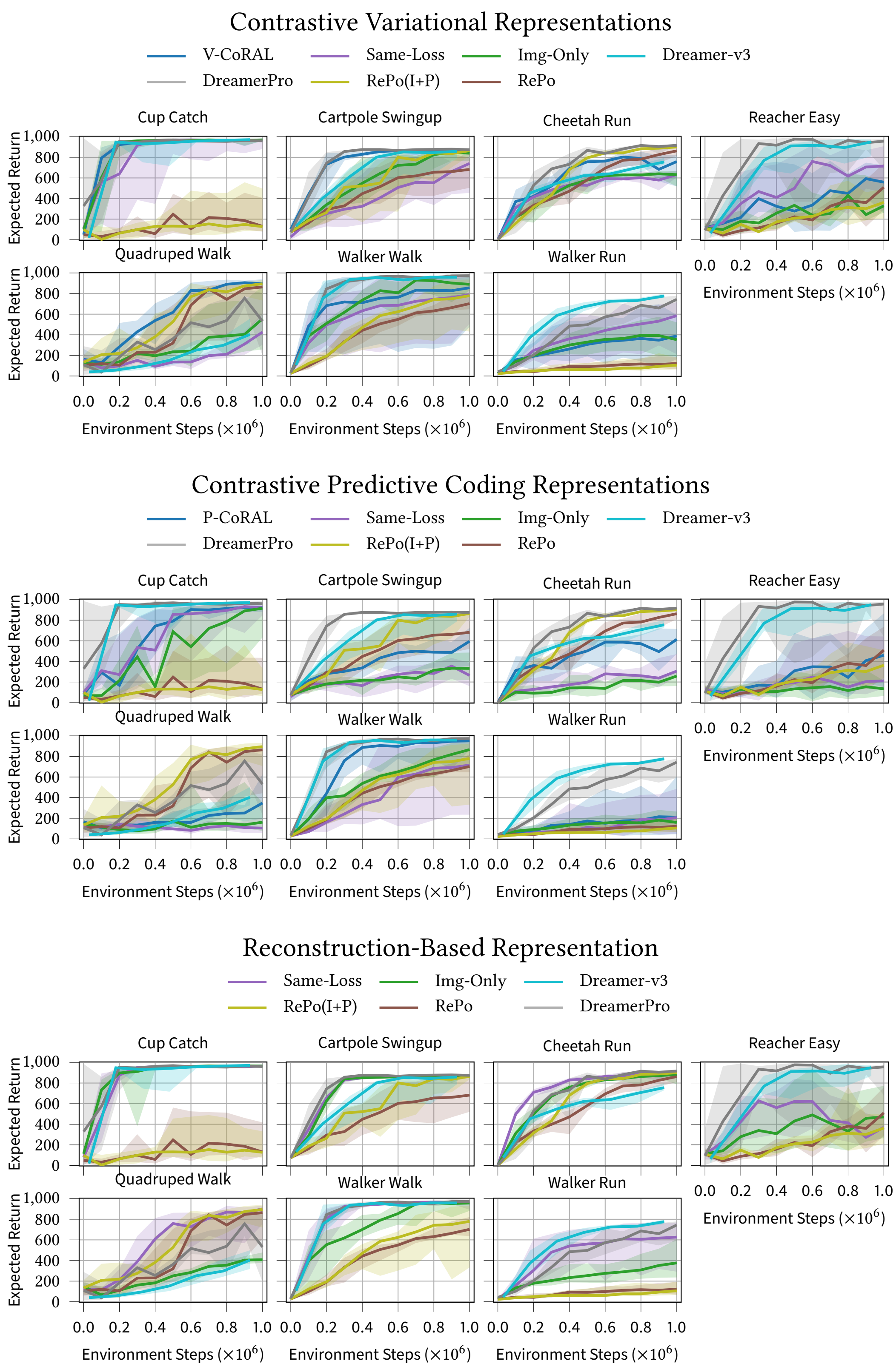


Figure E.10.: Per environment results for model-based agents on the DeepMind Control Suite with Standard Images.

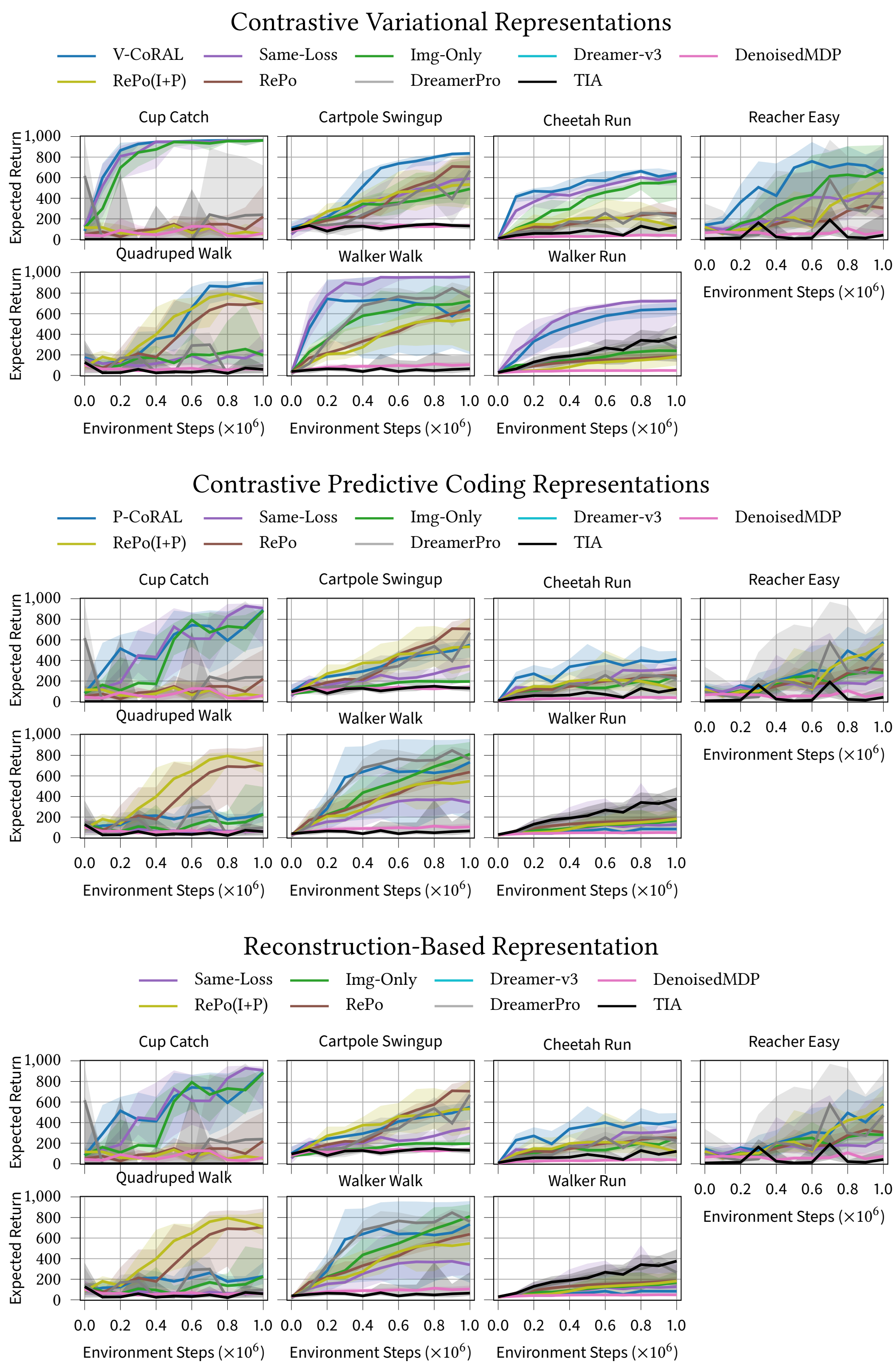


Figure E.11.: Per environment results for model-based agents on the DeepMind Control Suite with Video Background.

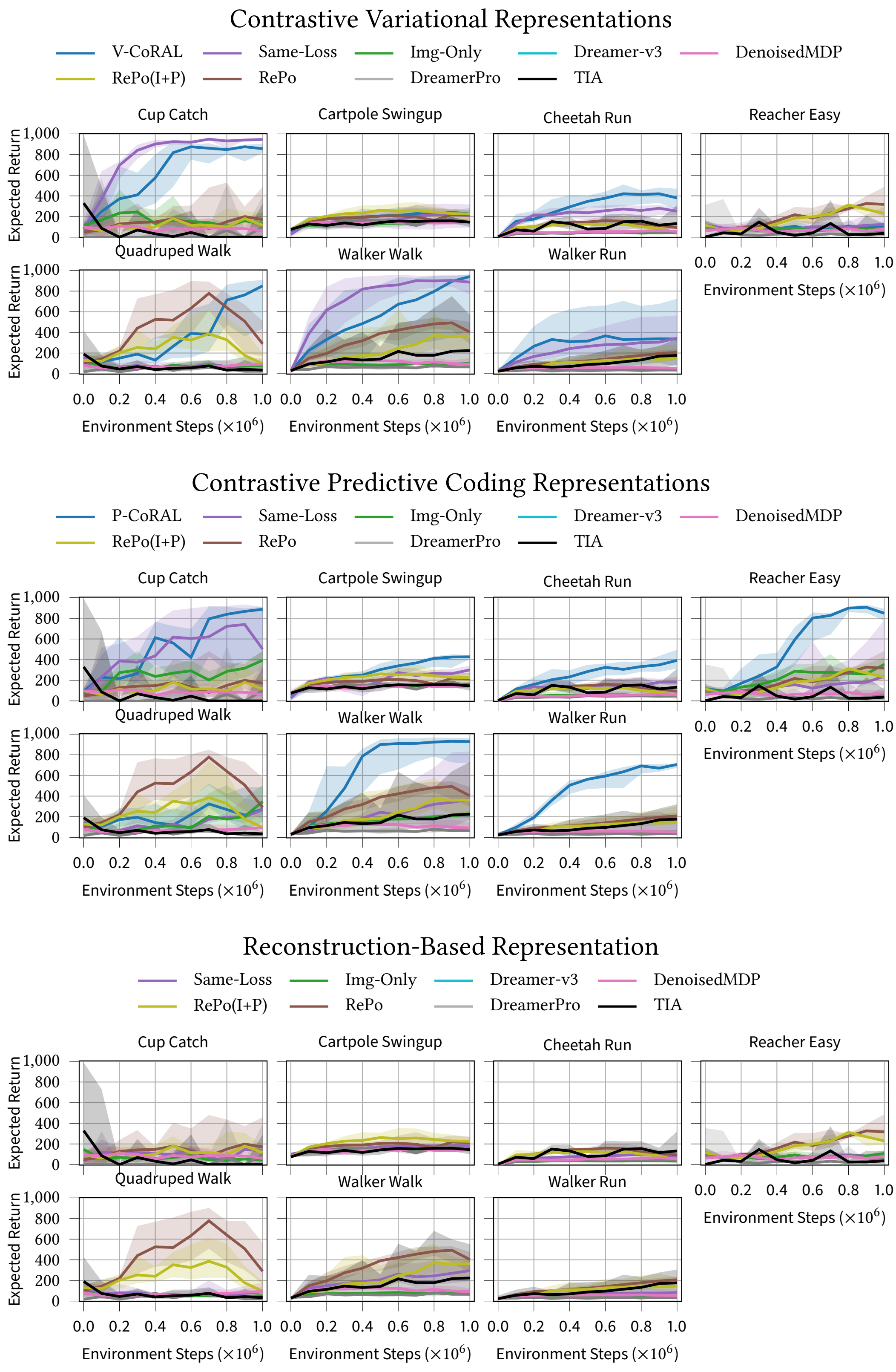


Figure E.12.: Per environment results for model-based agents on the DeepMind Control Suite with Occlusions.

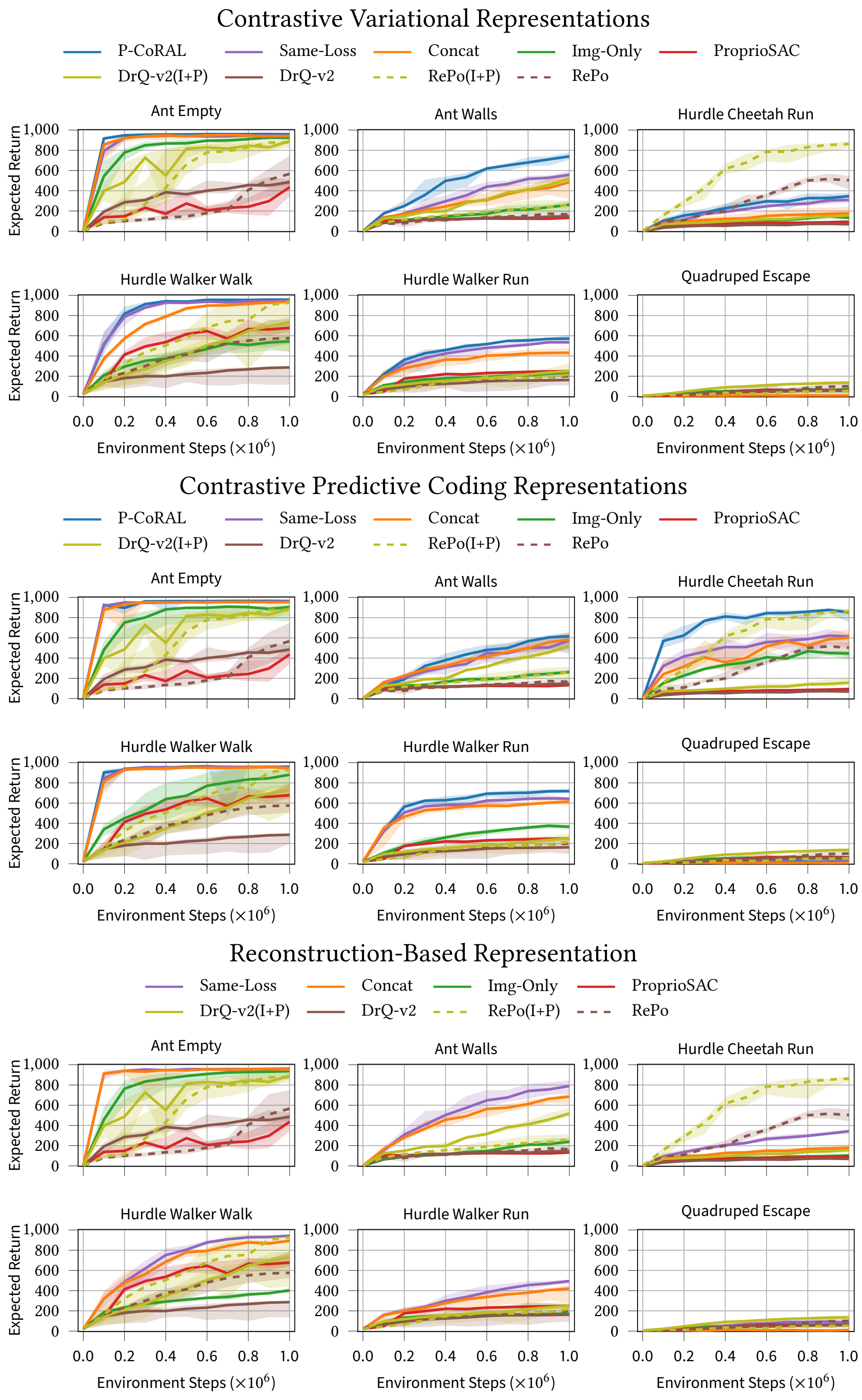


Figure E.13.: Per environment results for the Locomotion suite.

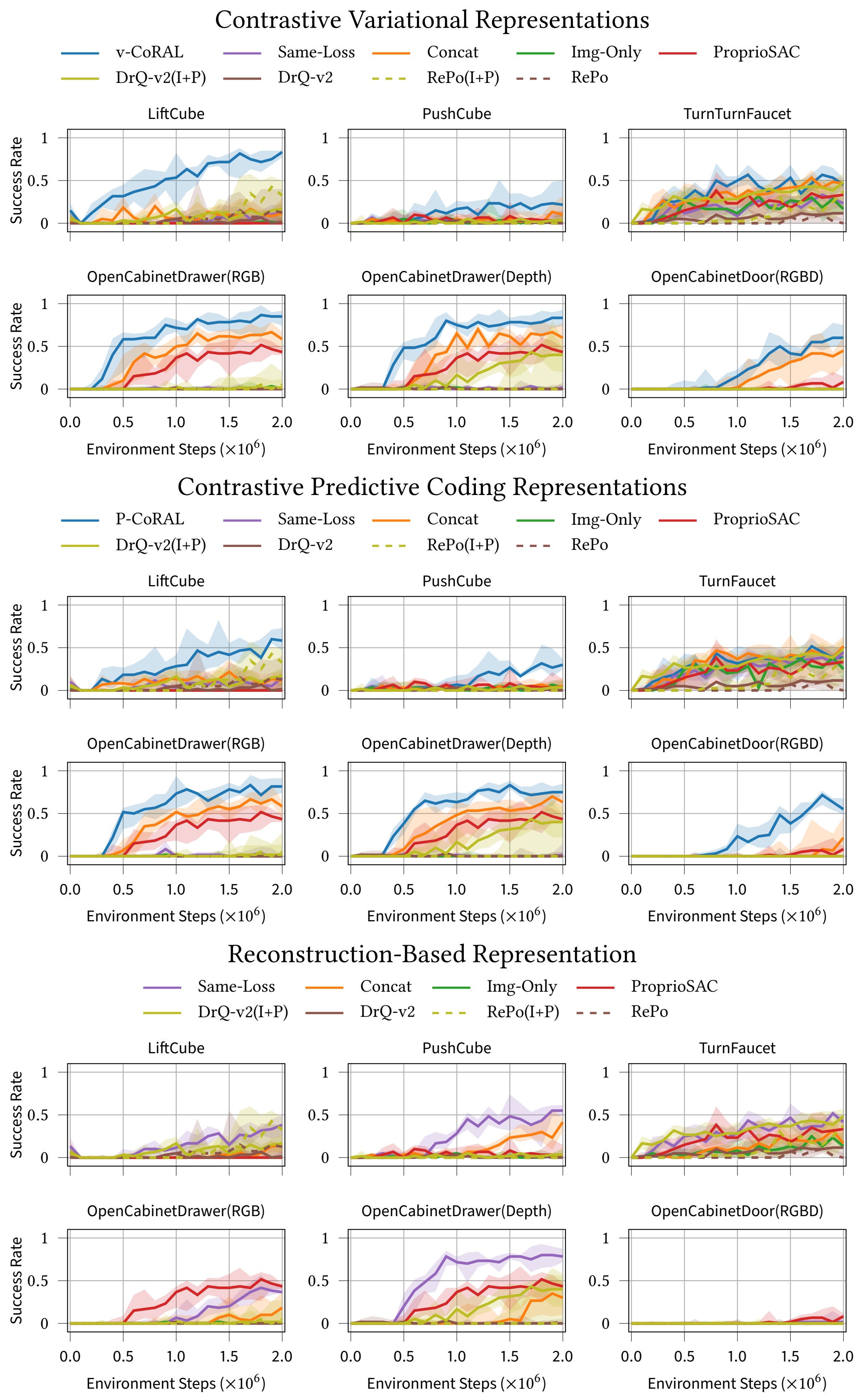


Figure E.14.: Per environment results for the Manipulation suite.