Bayesian Meta-Learning for Probabilistic Modeling and Optimization

Zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften

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

genehmigte

Dissertation

von

Michael Volpp

Tag der mündlichen Prüfung:	27. November 2024
1. Referent:	Prof. Dr. Gerhard Neumann
2. Referent:	Prof. Dr. Richard E. Turner

Abstract

Despite their proven potential, modern machine learning methods are often not readily applicable to practical settings, such as many modeling or optimization problems that arise in engineering and science. The reason is that it is usually impossible or too expensive to collect a sufficient amount of data to apply expressive modeling techniques or to find sufficiently accurate optimization solutions. Frequently, however, a large amount of historical data collected over time by solving similar problem instances ("tasks") is available. Meta-learning is a promising framework for conditioning machine learning algorithms on such data to solve novel tasks in a more data efficient manner.

Given the scarcity of data for novel tasks, the success of a meta-learning approach depends on an accurate quantification of the resulting predictive uncertainty, for which modern algorithms employ the paradigm of Bayesian inference. The resulting computations typically require sophisticated numerical approximation techniques to become efficient enough for practical applications. In this work, we show that state-of-the-art approaches for Bayesian meta-learning are often suboptimal in terms of the accuracy of these approximate Bayesian inferences. As a result, the predictive accuracy and uncertainty estimates do not meet the requirements for practical applications of meta-learning.

To bridge this gap, we develop novel algorithms that improve the performance of Bayesian meta-models and meta-optimizers without sacrificing computational efficiency. We first focus on the Bayesian modeling aspect and propose Bayesian Context Aggregation (BA) and Gaussian Mixture Neural Processes (GMM-NP) to achieve higher quality approximate Bayesian inferences. We show that these techniques not only significantly improve predictive performance, but also enable more efficient and accurate solutions to expensive optimization problems compared to the state of the art. In the second part of this work, we focus exclusively on such optimization problems and develop MetaBO, a novel algorithm that accelerates Bayesian optimization by directly meta-learning its optimization strategy.

Kurzfassung

Trotz ihres erwiesenen Potenzials sind moderne Methoden des maschinellen Lernens in der Praxis oft nicht ohne Weiteres anwendbar, z.B. für viele Modellierungs- oder Optimierungsprobleme, die in Technik und Wissenschaft auftreten. Dies liegt daran, dass es sich in der Regel als unmöglich oder zu teuer erweist, eine ausreichende Menge an Daten zu sammeln, um aussagekräftige Modellierungsverfahren anzuwenden oder hinreichend genaue Optimierungslösungen zu finden. Häufig ist jedoch eine große Menge an historischen Daten verfügbar, die im Laufe der Zeit durch die Lösung ähnlicher Problemstellungen („Aufgaben“) gesammelt wurden. Das Meta-Lernen ist ein vielversprechender Ansatz für die Konditionierung von Algorithmen des maschinellen Lernens auf solche Daten, um neue Aufgaben dateneffizienter zu lösen.

Angesichts der Knappheit der Daten für neue Aufgaben hängt der Erfolg des Meta-Lernens von einer genauen Quantifizierung der resultierenden Vorhersageunsicherheit ab. Um dieses Problem zu lösen, verwenden moderne Algorithmen das Paradigma der Bayes'schen Inferenz. Die daraus resultierenden Berechnungen erfordern in der Regel ausgefeilte numerische Approximationstechniken, um für praktische Anwendungen effizient genug zu sein. In dieser Arbeit zeigen wir, dass moderne Ansätze des Bayes'schen Meta-Lernens in Bezug auf die Genauigkeit dieser approximativen Bayes'schen Inferenzen oft suboptimal sind. Dies hat zur Folge, dass die Vorhersagegenauigkeit und die Unsicherheitsschätzungen nicht den Anforderungen für praktische Anwendungen des Meta-Lernens entsprechen.

Um diese Lücke zu schließen, entwickeln wir neuartige Algorithmen, die die Leistungsfähigkeit von Bayes'schen Meta-Modellen und Meta-Optimierern verbessern, ohne deren Recheneffizienz zu stark zu beeinträchtigen. Wir konzentrieren uns zunächst auf den Aspekt der Bayes'schen Modellierung und schlagen die Bayes'sche Kontextaggregation (BA) und Gaussian Mixture Neural Processes (GMM-NP) vor, um qualitativ hochwertigere approximative Bayes'sche Inferenzen zu erzielen. Wir zeigen, dass diese Techniken nicht nur die Vorhersagequalität erheblich verbessern, sondern auch effizientere und genauere Lösungen für teure Optimierungsprobleme im Vergleich zum Stand der Technik ermöglichen. Im zweiten Teil dieser Arbeit konzentrieren wir uns ausschließlich auf solche Optimierungsprobleme und entwickeln MetaBO, einen neuartigen Algorithmus, der die Bayes'sche Optimierung durch direktes Meta-Lernen der Optimierungsstrategie beschleunigt.

Acknowledgements

First of all, I would like to thank my parents, Petra and Herbert. You have taught me “what is important, and what is *really* important” [Len08], and through your unconditional love and support you have played the biggest part in me achieving this work. A big thank you also goes to my grandma Margot. Thank you for being by my side all my life and for showing us all how to stay happy, open-minded and mentally flexible into (slightly) advanced age despite many a hurdle! To my brother Stefan, thank you for being – in short – my best friend. Thank you for your rock-solid support, even in times when I wasn’t in the best of moods, and most of all for the joy you bring into my life through music. A very special thank you of course goes to my dear Nike. I’ve known you for exactly as long as I’ve known my dissertation, but unlike the latter, I haven’t had enough of you! Through all the ups and downs, you were the closest – thank you for not running away long ago! Thanks also to Leo, a wonderful friend, bandmate (and sister-in-law, by the way). You’ve really livened up the last few years and I’m glad about our trusting friendship. The same goes for Uschi, not least our karaoke parties have made the stressful phases of the past few years much more bearable! And of course I want to thank Petra and Alfred, the dearest parents-in-law you could wish for. Thank you for cheering me on and for your understanding when I often had to end my Sunday visits early because I had a paper to write. Of course, I would also like to thank all the aunts and uncles, cousins, nieces and nephews [sic!] and all the other members of my dear, big family – you are great!

To my supervisor Geri, thank you for welcoming me to all the institutes you led during my PhD. Thank you for the relaxed and collegial atmosphere in the research group at KIT and for your helpful input at any time of the day, night or vacation, which helped me out of many a scientific and emotional impasse. Thanks to Chris for your support at the BCAI and for taking me by the hand, especially on the stressful road to the first paper, and for standing by my side until the last minute before nightly deadlines, even when the heating and lights in the office had long since been switched off. Thanks also to all my colleagues and co-authors, especially Kathrin, Stefan, Julia, Edgar, Lukas G., Frank H., Steffen, Tobias, Felix^2, Jan, and Ilya for the interesting and helpful discussions, for proofreading and for the emotional support. I would also like to thank all my fellow PhD students for the great time at KIT and BCAI, especially Lukas F., Max (for being even older than me), Andreas, Philipp^2, Onur (for the Raki supply), Bruce, Vaisakh, Niklas, Nic, Aleks, Fabian^2, Ning, Hadi, Alex, Carlos, and Amrutha. And of course I would like to thank my students, Kirsten, Jan, Fabian, Anushka, Lucas, Denis, Florian, and Christian for your great work and for letting me learn so much with you.

Last but not least, I would like to thank Richard Turner and Jan Stühmer for their willingness to serve as referee and examiner for this thesis.

In memory of Hong Linh Thai.

Danksagung

Zuerst möchte ich mich bei meinen Eltern, Petra und Herbert, bedanken. Ihr habt mir beigebracht, „was wichtig, und was *wirklich* wichtig“ [Len08] ist, und durch eure bedingungslose Liebe und euren Rückhalt habt ihr den größten Anteil daran, dass ich diese Arbeit zustande gebracht habe. Ein großes Dankeschön geht auch an meine Oma Margot. Danke, dass du mein ganzes Leben an meiner Seite warst und dass du uns allen zeigst, wie man trotz so mancher Hürde bis ins (leicht) fortgeschrittene Alter glücklich, weltoffen und geistig flexibel bleiben kann! Meinem Bruder Stefan danke ich dafür, dass du – kurz gesagt – mein bester Freund bist. Danke für deine felsenfeste Unterstützung auch in Zeiten, in denen ich nicht gerade die beste Laune hatte, und vor allem für die Freude, die du durch die Musik in mein Leben bringst. Ein ganz besonderer Dank geht natürlich an meine liebe Freundin Nike. Ich kenne dich jetzt exakt genau so lange wie meine Diss, aber im Gegensatz zu dieser habe ich noch lange nicht genug von dir! Du warst bei allen Höhen und Tiefen am nächsten dran – danke, dass du nicht schon längst davongelaufen bist! Danke auch an Leo, eine wunderbare Freundin, Bandkollegin (und nebenbei auch noch Schwägerin). Du hast mir die letzten Jahre mächtig aufgelockert und ich bin froh über unsere vertrauensvolle Freundschaft. Dasselbe gilt für Uschi, nicht zuletzt unsere Karaokepartys haben die stressigen Phasen der vergangenen Jahre deutlich erträglicher gemacht! Und natürlich danke ich Petra und Alfred, den liebsten Schwiegereltern, die man sich wünschen kann. Danke fürs Mitfiebern und für euer Verständnis, wenn ich die sonntäglichen Besuche oft zu früh beenden musste, weil ein Paper geschrieben werden wollte. Natürlich gilt mein Dank auch allen Tanten und Onkels, Cousinen und Cousins, Nichten und Neffen [sic!] und allen weiteren Mitgliedern meiner lieben, großen Familie – ihr seid großartig!

Meinem Doktorvater Geri danke ich für die Aufnahme in all die Institute, die du während meiner Doktorarbeit geleitet hast. Danke für die entspannte und kollegiale Atmosphäre in der Forschungsgruppe am KIT und für deinen hilfreichen Input zu jeder Tages-, Nacht- und Urlaubszeit, der mich aus so mancher wissenschaftlichen und emotionalen Sackgasse befreit hat. Danke an Chris, für deine Betreuung am BCAI und dass du mich besonders auf dem stressigen Weg zum ersten Paper an die Hand genommen hast und mir bis zur letzten Minute vor nächtlichen Deadlines zur Seite gestanden bist, selbst als Heizung und Licht im Büro schon längst abgeschaltet waren. Danke auch an all meine Kolleg*innen und Koautor*innen, allen voran Kathrin, Stefan, Julia, Edgar, Lukas G., Frank H., Steffen, Tobias, Felix^2, Jan und Ilya für die interessanten und hilfreichen Diskussionen, fürs Korrekturlesen und für die emotionale Unterstützung. Gleichermaßen gilt mein Dank allen meinen Mitdoktorand*innen für die tolle Zeit am KIT und am BCAI, insbesondere Lukas F., Max (dafür, dass du sogar noch älter bist als ich), Andreas, Philipp^2, Onur (für den Rakisupport), Bruce, Vaisakh, Niklas, Nic, Aleks, Fabian^2, Ning, Hadi, Alex, Carlos und

Amrutha. Und selbstverständlich danke ich meinen Student*innen, Kirsten, Jan, Fabian, Anushka, Lucas, Denis, Florian und Christian, für eure großartige Arbeit und dafür, dass ich so viel mit euch lernen durfte.

Zu guter Letzt gilt mein Dank Richard Turner und Jan Stühmer, die sich bereit erklärt haben, als Gutachter und Prüfer dieser Doktorarbeit zu fungieren.

In Gedenken an Hong Linh Thai.

Contents

Appendix

1 Introduction and Summary of Contributions

Driven by algorithmic advances and the availability of increasingly powerful GPU-based hardware, the field of machine learning has produced a wealth of impressive results in recent years [Ros62, Rum86, LeC89, Rad21, He16, Kri12, Sim15, Bro21, Kin13, Goo14, Rez15, Din17, Soh15a, Ho20, Rom22, Mni15a, Sil17, Jum21, Vas17, Dev19, Rad19, Bro20, Ram21, Tou23, Ope24, Bub23]. One particularly successful branch that has revolutionized machine learning over the past decade is *deep learning*, which uses *neural networks* (NNs) as powerful general-purpose computational models to learn from data [LeC15, Goo16, Bis23]. Modern NNs have an exceptionally large number of adjustable parameters to achieve the expressiveness necessary to solve many problems of practical interest with satisfactory accuracy, with recent *large language models* (LLMs) exceeding the trillion-parameter threshold [Ope24].

To train such models, vast amounts of training data are required [Den09, Gok19, Sch21, Raf20, Bro20, Rad21, Kap20]. For a single task, sufficiently large datasets are typically not available, which renders naive approaches that start from randomly initialized NNs futile. An effective approach to remedy this problem is *transfer learning* [Zhu21, Hui21, Hos22, Hu21], which uses models pre-trained on large and diverse datasets to exhibit inductive biases towards a given type of task in the form of general-purpose data representations [Ben13]. These representations can then be refined and combined in novel ways to solve an application-specific *target task* from a similar domain. *Fine-tuning* a model in this way requires learning relatively few new parameters and can therefore be achieved with little training data from the target task.

This approach has recently been successfully scaled, notably through the development of the *transformer* architecture [Vas17], resulting in *foundation models* that can be fine-tuned for a wide range of target tasks [Sut19, Bom21]. Such models are currently transforming many areas of machine learning, such as natural language processing [Rad19, Dev19, Bro20, Ope24] or vision [Dos21], and are also being recognized in other fields of engineering and science [Che21, Ngu23, Yu23, Zvy23, Aze24, Bod24]. Interestingly, recent LLMs, which represent a special type of foundation model, exhibit *emergent abilities* that allow them to learn novel tasks from a few *context examples* provided in the prompt, without the need for fine-tuning [Ram21, Rad19, Bro20, Wei22, Cab23, Gar23]. Such *few-shot learning* capabilities are an example of *meta-learning* [Osw23], where a model *learns how to learn* tasks quickly, enabling generalization to unseen tasks from little context data.

Few-shot learning is particularly attractive for many problems encountered in engineering and science. Typical problems include modeling the dynamics or steady-state behavior of technical or physical systems, as well as identifying or optimizing their parameters [Gal14, Vol17, Pre07, Sha16]. In such settings, data acquisition usually requires expensive real-world experiments, so that the datasets for a given task often consist of only a handful of examples, making it impossible to train NNs of reasonable expressiveness from scratch. Unfortunately, however, the few-shot behavior of current LLMs can be difficult to control and interpret [Bow23], they can be expensive to train and evaluate [Str19, Pat21, Bal24], and they do not provide reliable estimates of predictive uncertainty [Pap24], which still limits their applicability to scientific data.

More specialized approaches do not rely on the emergent abilities of LLMs, but explicitly enforce meta-learning by building the multitask structure of the learning problem into the model architecture or training algorithm [Sch87, Thr98, Vil05, Hui21, Hos22]. The resulting *meta-model* learns data representations tailored for few-shot learning, is equipped with some kind of *adaptation mechanism* that allows it to solve tasks of a given type from a few context examples, and is trained on a structured *meta-dataset* of many related tasks of similar type. To solve an unseen target task of the same type, the adaptation mechanism is simply evaluated on the available context dataset.

The meta-learning approach depends on the availability of a meta-dataset of tasks similar to the target task. Fortunately, the data available in engineering and science problems often has such a multitask structure because similar tasks must be solved repeatedly. For example, a family of similar modeling tasks arises when a particular type of technical system, such as a car engine, is constantly being developed further or exists in several related designs. Similarly, determining optimal settings for machining workpieces made of varying materials or under varying ambient conditions gives rise to a family of related optimization tasks. In such situations, the number of context data examples for any given task is relatively small, but the joint meta-dataset of examples from all tasks accumulated over time can be massive, allowing a powerful meta-model to be trained. The central goal of this work is to develop meta-learning approaches suitable for such modeling and optimization problems.

A common approach to meta-modeling is to model each task conditional on a *latent task descriptor*, so that the adaptation mechanism corresponds to an operation that computes the task descriptor from the context dataset. During a meta-training phase, the parameters of the meta-model are optimized on the meta-dataset for predictive performance. To make predictions for the target task at test time, the parameters are fixed and the target task descriptor is determined from the context data using the adaptation mechanism. The inductive biases learned during the meta-training phase then restrict the predictions to hypotheses compatible with the meta-data, while the target task descriptor determines which of these hypotheses is also compatible with the context data.

Due to *task ambiguity*, few-shot context datasets are typically compatible with a range of hypotheses, reflecting *epistemic uncertainty* about the task descriptor. A major focus of this work is the accurate quantification of this uncertainty, which is essential for many real-world applications [Pap24]. A principled treatment is enabled by adopting a *probabilistic modeling* approach and using *Bayesian inference* as the adaptation mechanism, which quantifies epistemic uncertainty in terms of a probability distribution over the space of task descriptors. Methods that compute such *task posterior distributions* define the field of *Bayesian meta-learning* (BML).

To obtain models of reasonable complexity, modern BML approaches parameterize the probabilistic meta-models using deep NNs [Hui21, Hos22]. In such models, task posterior inference is computationally intractable, so Bayesian inference requires approximations. In this context, the *neural process* (NP) [Gar18c] represents a particularly appealing NN-based parameterization, as it defines a comparatively low-dimensional task descriptor space, which allows to draw from a rich toolbox of approximate Bayesian inference techniques [Mac03, Gel13, Bis06]. In its vanilla form, the NP uses *amortized variational inference* [Kin13] to compute a *Gaussian mean-field* approximation of the task posterior distribution using an *encoder NN*. A *deep set* [Zah17] parameterization allows to feed context sets of variable sizes through this encoder, while preserving the invariance of the Bayesian inference problem to permutations of the context examples. In essence, this amounts to processing each context example individually using a shared NN and computing the task posterior distribution from the arithmetic mean of the resulting representations.

In Volpp et al. [Vol21], we show that this *mean aggregation* operation handles task ambiguity suboptimally, and propose a novel *Bayesian context aggregation* (BA) scheme. The central insight is to weight each context example according to its information content about the correct hypothesis. We achieve this by reinterpreting context aggregation as task descriptor inference, so that the weights emerge naturally as the solution to a Bayesian inference problem that can be computed efficiently from the context data. We demonstrate that BA improves the performance of NP-based BML models in few-shot learning scenarios, and subsequent work successfully applies it in domains such as computer vision [Gao22] and robotics [Li23].

As a deep set variant, BA still operates within the framework of amortized variational inference and computes a Gaussian mean-field task posterior approximation that is trained using *reparameterized Euclidean gradients* [Kin13]. In Volpp et al. [Vol23], we study the impact of these design choices and show that they can introduce significant approximation errors that degrade predictive performance. To mitigate this, we propose the *Gaussian mixture neural process* (GMM-NP), which computes an expressive GMM approximation of the task posterior distribution that captures multimodality and correlations. We show that this yields significantly improved predictions, and that the GMM approximations can be computed in a robust and efficient manner building on trust regions and natural gradients for variational optimization [Are23]. Recent work [Dah23] further demonstrates the effectiveness of our approach for few-shot learning material properties in mesh-based simulations.

To approach meta-optimization problems, we consider *Bayesian optimization*, a powerful algorithm for global optimization of objective functions that are expensive to evaluate [Sha16]. BO formulates an optimization strategy in terms of an *acquisition function* (AF) that trades off exploration of unknown regions of the search space against exploitation of regions known to yield high objective values. The AF is computed from the predictions of a probabilistic *surrogate model* of the history of objective function evaluations, so the effectiveness of BO depends critically on well-calibrated uncertainty estimates. In Volpp et al. [Vol23], we show that replacing this surrogate model with our meta-models yields global meta-optimizers that can efficiently solve expensive optimization problems, underscoring the quality of the epistemic uncertainty estimates that result from our improved approximate inference techniques.

This approach to meta-optimization leaves the AF of BO untouched and injects inductive biases towards the given type of optimization problem through a meta-surrogate model. In Volpp et al. [Vol20], we develop *MetaBO*, a complementary approach that does not modify the probabilistic surrogate model and instead meta-learns the AF. To this end, we represent the AF as a NN that computes the next evaluation location from the probabilistic predictions of a standard surrogate model. To induce inductive biases towards the given type of optimization problem, we train this *neural AF* to maximize optimization performance on the meta-dataset using *reinforcement learning* (RL) [Sut18], a powerful method for optimizing sequential decision-making problems such as the optimization strategy of BO.

In addition to the main articles described above, student theses supervised in the context of this work explore several other aspects of the BML problem, namely *meta-overfitting* [Vas21], the accurate evaluation of Bayesian meta-models [Flo22], and stable gradient-based optimization of Gaussian observation models [Meg22]. Furthermore, two co-authored research articles study problems in the related fields of Bayesian deep learning [Sel23] and RL [Doe19].

The organization of this work is as follows. In Sec. 2.1, we develop the theoretical background necessary for understanding Bayesian meta-learning. In particular, we motivate Bayesian inference as a powerful paradigm for generalizing from small datasets, study the asymptotic behavior of Bayesian inferences, discuss various sources of uncertainty that arise in general learning problems, derive modern approximate inference techniques, and introduce BO. In Sec. 2.2, this groundwork allows us to derive a Bayesian meta-model architecture proposed by Heskes [Hes00] and Bakker et al. [Bak03] which forms the basis of the NP [Gar18c], and to introduce several alternative meta-learning approaches. Building on these ideas, we then develop our novel meta-learning algorithms, Bayesian context aggregation in Sec. 3, GMM-NP in Sec. 4, and MetaBO in Sec. 5. In Sec. 6 we summarize our work and assess possible avenues for future research.

2 Theoretical Background

We now summarize the theoretical background necessary for the methods developed in this work. First, we review general concepts from Bayesian inference with a focus on modern approximation schemes. We then introduce Bayesian optimization, as well as a generic Bayesian multitask model from which many current meta-learning approaches are derived, and present the neural process [Gar18c], a specific instantiation of this model. We conclude the chapter with an overview of related meta-learning methods.

2.1 Bayesian Inference

In this section, we review Bayesian inference, the statistical framework underlying Bayesian meta-learning.

2.1.1 Single-Task Datasets

We consider a function $f : \mathcal{X} \to \mathcal{Y}$, where $\mathcal{X} \subset \mathbb{R}^{d_x}$ and $\mathcal{Y} \subset \mathbb{R}^{d_y}$ with $d_x, d_y \in \mathbb{N}$. f is assumed to be *black-box*, in the sense that we can obtain information about it only by collecting evaluations at arbitrary input locations in $\mathcal{X}$. From a dataset of such evaluations, we aim to *infer a model* that summarizes our knowledge about f, a statistical method called *regression* [Bis06, Mur23, Mac03].

Specifically, we consider a dataset $\mathcal{D}^c$ of *inputs* $\boldsymbol{x}_n^c \in \mathcal{X}$ and corresponding *targets* $\boldsymbol{y}_n^c \in \mathcal{Y}$,

$$\mathcal{D}^c = \left\{ (\boldsymbol{x}_n^c, \boldsymbol{y}_n^c) \in \mathcal{X} \times \mathcal{Y} \mid n \in \{1, \dots, N^c\} \right\}, \tag{2.1}$$

which we assume to be generated by noise-corrupted evaluations of f, i.e.,[1]

$$\boldsymbol{x}_n^c \sim p^{\mathcal{D}}(\boldsymbol{x}), \quad \boldsymbol{\varepsilon}_n^c \sim p^{\mathcal{D}}(\boldsymbol{\varepsilon} \mid \boldsymbol{x}_n^c), \quad \boldsymbol{y}_n^c = f(\boldsymbol{x}_n^c) + \boldsymbol{\varepsilon}_n^c. \tag{2.2}$$

[1] In this work, we follow the common convention in the field of machine learning to formulate problems in terms of random variables and their probability density functions, leaving the definitions of the underlying sample space and probability measure implicit [Was04]. Furthermore, to unclutter notation, we often use the same symbol (e.g., p or $p^{\mathcal{D}}$) to denote different probability density functions and distinguish them by "dummy" arguments. Similarly, our notation does not distinguish between random variables and their realizations, since the interpretation is usually clear from the context [Bis06, Gel13]. We also use the terms "probability distribution", "probability density", and "probability density function" interchangeably.

The inputs $\boldsymbol{x}_n^c$ are distributed according to a probability density $p^{\mathcal{D}}(\boldsymbol{x})$ on $\mathcal{X}$, and the random variables $\boldsymbol{\varepsilon}_n^c$ with density $p^{\mathcal{D}}(\boldsymbol{\varepsilon} \mid \boldsymbol{x}_n^c)$ describe an additive noise process on $\mathcal{Y}$. In general, the noise may be *heteroscedastic*, i.e., the density $p^{\mathcal{D}}(\boldsymbol{\varepsilon} \mid \boldsymbol{x})$ may depend on the input location. Note that we use the superscript c to indicate the correspondence to what we call a context dataset in meta-learning.

Following Eq. (2.2), we assume that the data tuples $(\boldsymbol{x}_n^c, \boldsymbol{y}_n^c)$ that make up $\mathcal{D}^c$ are sampled *independently and identically* (i.i.d.) from a joint distribution

$$p^{\mathcal{D}}(\boldsymbol{x}, \boldsymbol{y}) = p^{\mathcal{D}}(\boldsymbol{y} \mid \boldsymbol{x})\, p^{\mathcal{D}}(\boldsymbol{x}), \tag{2.3}$$

defined by the distributions $p^{\mathcal{D}}(\boldsymbol{x})$ and $p^{\mathcal{D}}(\boldsymbol{y} \mid \boldsymbol{x})$. Note that $p^{\mathcal{D}}(\boldsymbol{y} \mid \boldsymbol{x})$ is defined implicitly by Eq. (2.2) through the function f and the additive noise process. We call $p^{\mathcal{D}}(\boldsymbol{x}, \boldsymbol{y})$ the *data distribution* of the regression problem we aim to solve. In the following, we will refer to one such regression problem as a *task*. Since there is only one regression problem involved in our current setting, we call it a *single-task problem* to distinguish it from the multitask problems we will study later in this work.

Our goal is to use $\mathcal{D}^c$ as a *training dataset* to infer a model that *generalizes* to an unobserved *test dataset*

$$\mathcal{D}^t = \left\{ \left(\boldsymbol{x}_m^t, \boldsymbol{y}_m^t\right) \in \mathcal{X} \times \mathcal{Y} \mid m \in \left\{1, \dots, N^t\right\} \right\}, \tag{2.4}$$

that was not used for training, i.e., $\mathcal{D}^c \cap \mathcal{D}^t = \varnothing$. The underlying assumption that makes this approach reasonable is that the training and test datasets share statistical structure, in the sense that observing the training data provides information about the generative process of the test data. Typically, we assume that the training and test datasets are identically distributed, i.e., that $(\boldsymbol{x}_m^t, \boldsymbol{y}_m^t) \sim p^{\mathcal{D}}(\boldsymbol{x}, \boldsymbol{y})$ for $m \in \{1, \dots, N^t\}$.

2.1.2 Bayesian Modeling

Let us now consider model inference, given data of the structure introduced in the previous section. Since inferences are inherently uncertain due to noise and finite training data, we use a *probabilistic modeling* approach that allows for a rigorous mathematical treatment of this uncertainty [Was04, Gel13, Bis06, Hül21]. The assumption of i.i.d. data motivates its description in terms of a *parametric model*, i.e., a family of conditional probability distributions[1]

$$\{p(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{z}) \mid \boldsymbol{z} \in \mathcal{Z}\}, \tag{2.5}$$

that describe the data as *conditionally independent* given $\boldsymbol{z}$ [Fin30, Fin37, Hew55, Kin78, Ric06, Gel13]. We call the set $\mathcal{Z} \subset \mathbb{R}^{d_z}$ the *hypothesis space* and each $\boldsymbol{z} \in \mathcal{Z}$ a *hypothesis*.[2]

[1] Note that we do not model the distribution $p^{\mathcal{D}}(\boldsymbol{x})$ of the inputs.

[2] We will use the terms “hypothesis” and “parameter” interchangeably throughout this work.

A major focus of this work is generalization from *small* training datasets $\mathcal{D}^c$, in the sense that $\mathcal{D}^c$ is compatible with many different hypotheses $\boldsymbol{z} \in \mathcal{Z}$. In such cases, it is nonsensical to settle on a *point estimate*, i.e., a single hypothesis $\hat{\boldsymbol{z}} \in \mathcal{Z}$, to describe the data. In fact, for reasonably flexible models, this approach bears the risk of *overfitting*, where $p(\boldsymbol{y} \mid \boldsymbol{x}, \hat{\boldsymbol{z}})$ explains $\mathcal{D}^c$ well but generalizes poorly to the test dataset $\mathcal{D}^t$. To avoid overfitting, we follow a *Bayesian approach to generalization*, which uses a *Bayesian model average* (BMA) [Mac03, Bis06, Gel13, Wil20, Pap24], i.e., a weighted combination of predictions with *all* hypotheses in $\mathcal{Z}$, with higher weights given to hypotheses that better explain the training data. In this way, we express and quantify the predictive uncertainty arising due to the lack of training data, rather than relying on a single and likely wrong hypothesis. Intuitively, high predictive uncertainty is expressed by a BMA that assigns significant weights to several conflicting hypotheses, while a BMA that contains only consistent hypotheses indicates low predictive uncertainty.

Specifically, a Bayesian approach computes a *reweighting* of hypotheses in $\mathcal{Z}$, starting from an initial weighting chosen according to our *prior belief* about the plausibility of hypotheses. This belief can be expressed formally in terms of a probability distribution $p(\boldsymbol{z})$ over $\mathcal{Z}$ [Cox46, Cox61]. In this way, $\boldsymbol{z}$ is promoted to a *latent* (i.e., unobserved) *random variable*, and Bayesian predictions follow by a straightforward application of the rules of probability theory [Mac03, Bis06, Gel13]. We will refer to the parametric model $\{p(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{z}) \mid \boldsymbol{z} \in \mathcal{Z}\}$ together with the specification of the prior distribution $p(\boldsymbol{z})$ as the *Bayesian model*, or simply as the *model*.

Our conditional independence assumptions are summarized in the probabilistic graphical model [Kol09, Bis06] Fig. 2.1. The joint distribution over all unobserved variables factorizes as

$$p\left(\boldsymbol{y}^t_{1:N^t}, \boldsymbol{z} \mid \boldsymbol{x}^t_{1:N^t}, \mathcal{D}^c\right) = \prod_{m=1}^{N^t} p\left(\boldsymbol{y}^t_m \mid \boldsymbol{x}^t_m, \boldsymbol{z}\right) p\left(\boldsymbol{z} \mid \mathcal{D}^c\right), \tag{2.6}$$

where we have introduced the shorthand notation $\boldsymbol{y}^t_{1:N^t} \equiv \left\{\boldsymbol{y}^t_m \in \mathcal{Y} \mid m \in \left\{1, \dots, N^t\right\}\right\}$, which we also use for other quantities such as $\boldsymbol{x}^t_{1:N^t}$. Marginalizing this distribution over hypotheses yields Bayesian predictions in terms of a *predictive distribution.* In the absence of training data, i.e., for $\mathcal{D}^c = \varnothing$, we obtain the *prior predictive distribution* as a superposition of $p(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{z})$, weighted by the corresponding prior probability densities, i.e.,

$$p\left(\boldsymbol{y}^t_{1:N^t} \mid \boldsymbol{x}^t_{1:N^t}\right) = \int \prod_{m=1}^{N^t} p\left(\boldsymbol{y}^t_m \mid \boldsymbol{x}^t_m, \boldsymbol{z}\right) p(\boldsymbol{z}) \, \mathrm{d}\boldsymbol{z}. \tag{2.7}$$

Similarly, in the case $\mathcal{D}^c \neq \varnothing$, we obtain the *posterior predictive distribution*

$$p\left(\boldsymbol{y}^t_{1:N^t} \mid \boldsymbol{x}^t_{1:N^t}, \mathcal{D}^c\right) = \int \prod_{m=1}^{N^t} p\left(\boldsymbol{y}^t_m \mid \boldsymbol{x}^t_m, \boldsymbol{z}\right) p\left(\boldsymbol{z} \mid \mathcal{D}^c\right) \, \mathrm{d}\boldsymbol{z}. \tag{2.8}$$

Comparing this expression with Eq. (2.7), we observe that hypotheses have been reweighted in the sense that they are now multiplied by the *posterior probability densities* $p\left(\boldsymbol{z} \mid \mathcal{D}^c\right)$.

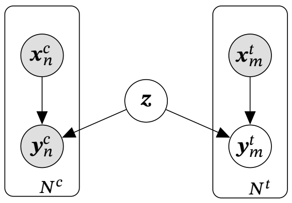


Figure 2.1: Probabilistic graphical model [Kol09] derived from the assumption of independent and identically distributed training data (left plate) and test data (right plate) [Kin78]. It defines the joint probability distribution over unobserved variables $\boldsymbol{y}_{1:N^t}^t$ and $\boldsymbol{z}$ in terms of a parametric model $\{p(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{z}) \mid \boldsymbol{z} \in \mathcal{Z}\}$ and a prior distribution $p(\boldsymbol{z})$ over hypotheses.

For any fixed training set $\mathcal{D}^c$, Eq. (2.8) defines a set of probability distributions over the finite-dimensional[1] space $\mathcal{Y}^{N^t}$, parameterized by the inputs $\boldsymbol{x}_{1:N^t}^t$. Note that this set of distributions is consistent in the sense that it is invariant with respect to permutations of the target tuples $(\boldsymbol{x}_m^t, \boldsymbol{y}_m^t)$, and that marginalizing out a subset of the variables $\boldsymbol{y}_{1:N^t}^t$ yields the distribution over the corresponding subspace. Under these consistency conditions, the *Kolmogorov extension theorem* asserts the existence of a unique probability distribution over the infinite-dimensional space of *functions* $\mathcal{X} \to \mathcal{Y}$ (formalized by the concept of a *stochastic process*) that has the above distributions as its finite-dimensional marginal distributions [Øks10, Gar18c, Foo20a]. Thus, we can consider this stochastic process as being defined by the finite-dimensional marginal distributions Eq. (2.8).

The posterior distribution $p(\boldsymbol{z} \mid \mathcal{D}^c)$ formally expresses our *posterior belief* about the corresponding hypotheses, obtained by updating our prior belief with the information about the correct hypothesis contained in the training data $\mathcal{D}^c$. This update is given by *Bayes' theorem* [Bay63, Cox61], which expresses the posterior distribution as

$$p(\boldsymbol{z} \mid \mathcal{D}^c) = \frac{p\left(\boldsymbol{y}_{1:N^c}^c \mid \boldsymbol{x}_{1:N^c}^c, \boldsymbol{z}\right) p(\boldsymbol{z})}{p\left(\boldsymbol{y}_{1:N^c}^c \mid \boldsymbol{x}_{1:N^c}^c\right)}. \tag{2.9}$$

The quantity $p\left(\boldsymbol{y}_{1:N^c}^c \mid \boldsymbol{x}_{1:N^c}^c, \boldsymbol{z}\right)$, read as a function of the hypothesis $\boldsymbol{z}$, is called the *likelihood function*, and its value at $\boldsymbol{z}$ is called the *likelihood* of $\boldsymbol{z}$. Under our model, the likelihood function factorizes as

$$p\left(\boldsymbol{y}_{1:N^c}^c \mid \boldsymbol{x}_{1:N^c}^c, \boldsymbol{z}\right) = \prod_{n=1}^{N^c} p\left(\boldsymbol{y}_n^c \mid \boldsymbol{x}_n^c, \boldsymbol{z}\right). \tag{2.10}$$

The normalizing factor in the denominator is called the *marginal likelihood*, or *evidence*,

$$p\left(\boldsymbol{y}_{1:N^c}^c \mid \boldsymbol{x}_{1:N^c}^c\right) = \int \prod_{n=1}^{N^c} p\left(\boldsymbol{y}_n^c \mid \boldsymbol{x}_n^c, \boldsymbol{z}\right) p(\boldsymbol{z}) \, \mathrm{d}\boldsymbol{z}. \tag{2.11}$$

[1] For the sake of this discussion, let $\mathcal{Y}$ denote the vector space $\mathbb{R}^{d_y}$.

Intuitively, the marginal likelihood describes the probability of generating the training data $\boldsymbol{y}^c_{1:N^c}$, if we were to sample randomly from the prior distribution $p(\boldsymbol{z})$ [Mac03, Wil20].[1]

2.1.3 Asymptotic Behavior of Bayesian Inferences

In the previous section, we motivated a Bayesian modeling approach for a small training dataset $\mathcal{D}^c$ as a method to avoid overfitting. This involves quantifying our belief about the degree of compatibility of hypotheses $\boldsymbol{z} \in \mathcal{Z}$ with $\mathcal{D}^c$ in terms of a posterior distribution $p(\boldsymbol{z} \mid \mathcal{D}^c)$, and incorporating this belief into predictions using the predictive distribution Eq. (2.8). We will now examine how Bayesian predictions behave in the complementary setting of a large training dataset, i.e., we consider the asymptotic behavior of $p(\boldsymbol{z} \mid \mathcal{D}^c)$ in the limit $N^c \to \infty$. We will find that in this limit the posterior distribution is well approximated by a point estimate, reflecting the intuition that we should be able to determine the best hypothesis with certainty when we have infinite training data.

Our considerations will also reveal in which sense and in which situations this point estimate is asymptotically correct. To make this precise, we shall distinguish the *well-specified* case, where the data distribution belongs to the parametric model, i.e., where there exists a "true" hypothesis $\boldsymbol{z} \in \mathcal{Z}$ with $p^{\mathcal{D}}(\boldsymbol{y} \mid \boldsymbol{x}) = p(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{z})$, from the *misspecified* case, where there is no such hypothesis in $\mathcal{Z}$. We will see that Bayesian inferences asymptotically yield the true hypothesis in the well-specified case, and "come as close as possible" (in a sense we will formalize below) in the misspecified case. We will keep the discussion relatively informal and refer the interested reader to the literature for further details and proofs [Lap10, Doo49, Cam53, Vaa00, Was04, Ric06, Kle12, Gel13, Gey13, Fan16].

Let us first introduce the *Kullback-Leibler (KL) divergence* [Kul51], which is defined for probability densities p and q over the same space as

$$\mathrm{KL}\left[p \parallel q\right] = \mathbb{E}_{p(\boldsymbol{y})}\left[\log \frac{p(\boldsymbol{y})}{q(\boldsymbol{y})}\right]. \tag{2.12}$$

The KL divergence is an example of a *statistical divergence* [Ama16]. In particular, it is positive definite, i.e., for any p and q it holds that $\mathrm{KL}\left[p \parallel q\right] \geq 0$, with equality if and only if $p = q$.

In the following, we assume that there exists a unique hypothesis $\bar{\boldsymbol{z}} \in \mathcal{Z}$ that minimizes the KL divergence between the data distribution and the model, i.e.,

$$\bar{\boldsymbol{z}} = \arg\min_{\boldsymbol{z} \in \mathcal{Z}} \mathbb{E}_{p^{\mathcal{D}}(\boldsymbol{x})}\left\{\mathrm{KL}\left[p^{\mathcal{D}}(\cdot \mid \boldsymbol{x}) \parallel p(\cdot \mid \boldsymbol{x}, \boldsymbol{z})\right]\right\}. \tag{2.13}$$

[1] Note that due to our notation conventions described in Sec. 2.1.1, the expression for the marginal likelihood Eq. (2.11) appears equivalent in structure to the prior predictive distribution Eq. (2.7). However, these quantities are conceptually very different. Eq. (2.7) describes the joint probability density function of unobserved function values $\boldsymbol{y}^t_{1:N^t}$, which arises as the marginal density of the underlying stochastic process at locations $\boldsymbol{x}^t_{1:N^t}$. In contrast, Eq. (2.11) is the real number obtained by evaluating the marginal density at locations $\boldsymbol{x}^c_{1:N^c}$ on the observed data $\boldsymbol{y}^c_{1:N^c}$.

Since we are not modeling the input distribution $p^{\mathcal{D}}(\boldsymbol{x})$, both the data and the model distributions come with the same factor $p^{\mathcal{D}}(\boldsymbol{x})$, which canceled out in the fraction in Eq. (2.12). $p(\boldsymbol{y} \mid \boldsymbol{x}, \bar{\boldsymbol{z}})$ is called the *forward KL projection* of the data distribution onto the parametric model. In the well-specified case, $\bar{\boldsymbol{z}}$ coincides with the true hypothesis (by positive definiteness of the KL divergence), which we will thus also denote by $\bar{\boldsymbol{z}}$. In the misspecified case, such a true hypothesis does not exist, and $\bar{\boldsymbol{z}}$ can be interpreted as the hypothesis that brings the distribution $p(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{z})$ closest to the data distribution in terms of KL divergence.

We can now state the *Bernstein-von Mises theorem* [Doo49], which is the central result describing the asymptotic behavior of Bayesian inferences. It asserts that, under suitable regularity conditions, the posterior distribution $p(\boldsymbol{z} \mid \mathcal{D}^c)$ asymptotically approaches a Normal distribution with mean $\bar{\boldsymbol{z}}$.[1] An important condition for this to hold is that the prior distribution must be supported at $\bar{\boldsymbol{z}}$, i.e., $p(\bar{\boldsymbol{z}}) > 0$. Furthermore, the theorem asserts that Bayesian inferences are *asymptotically certain*, i.e., the limiting Normal distribution (and, thus, the posterior distribution) becomes more and more concentrated around $\bar{\boldsymbol{z}}$ as the training dataset $\mathcal{D}^c$ grows.

We gain further insight into the role of $\bar{\boldsymbol{z}}$ using a similar result from frequentist statistics about the *maximum likelihood estimator* (MLE),

$$\hat{\boldsymbol{z}} = \arg\max_{\boldsymbol{z} \in \mathcal{Z}} \frac{1}{N^c} \sum_{n=1}^{N^c} \log p(\boldsymbol{y}_n^c \mid \boldsymbol{x}_n^c, \boldsymbol{z}), \tag{2.14}$$

which states that, under suitable regularity conditions, the MLE is *consistent* for $\bar{\boldsymbol{z}}$, i.e., its sampling distribution becomes more and more concentrated around $\bar{\boldsymbol{z}}$ for large $\mathcal{D}^c$ [Was04]. In this sense, Bayesian and frequentist inferences are equivalent for large training datasets. Moreover, this observation implies *asymptotic consensus*, i.e., that Bayesian predictions asymptotically become independent of the prior distribution $p(\boldsymbol{z})$. In fact, Eq. (2.14) is independent of $p(\boldsymbol{z})$, so the consistency of the MLE for $\bar{\boldsymbol{z}}$ implies the asymptotic independence of $p(\boldsymbol{z} \mid \mathcal{D}^c)$ from the prior distribution.

These results show that for large training datasets it is meaningful to approximate the posterior distribution with a point estimate at $\hat{\boldsymbol{z}}$, i.e.,

$$p(\boldsymbol{z} \mid \mathcal{D}^c) \approx \delta(\boldsymbol{z} - \hat{\boldsymbol{z}}) \quad \text{for} \quad N^c \to \infty. \tag{2.15}$$

Here, $\delta(\boldsymbol{z})$ denotes the Dirac delta distribution [Dir30, Sch50]. Using Eq. (2.15) in the predictive distribution Eq. (2.8), yields the *plugin approximation* [Mur23] of the predictive distribution

$$p\left(\boldsymbol{y}_{1:N^t}^t \mid \boldsymbol{x}_{1:N^t}^t, \mathcal{D}^c\right) \approx \prod_{m=1}^{N^t} p\left(\boldsymbol{y}_m^t \mid \boldsymbol{x}_m^t, \hat{\boldsymbol{z}}\right) \quad \text{for} \quad N^c \to \infty. \tag{2.16}$$

[1] The convergence is in probability with respect to the data distribution $p^{\mathcal{D}}(\boldsymbol{x}, \boldsymbol{y})$, using the total variation as the distance measure between the posterior distribution and the limiting Normal distribution.

In summary, for large training datasets, Bayesian predictions are well approximated by predictions with the single "best" hypothesis $\bar{\boldsymbol{z}} \in \mathcal{Z}$, which can be estimated from the training data $\mathcal{D}^c$ by $\hat{\boldsymbol{z}}$.

2.1.4 Sources of Uncertainty

We now examine various sources of uncertainty that arise in general learning problems and how these uncertainties are quantified in Bayesian modeling [Hül21]. First, assume that the ground truth data generating process $p^{\mathcal{D}}(\boldsymbol{y} \mid \boldsymbol{x})$ was known. Since the targets $\boldsymbol{y}$ are corrupted by noise according to Eq. (2.2), the dependence of $\boldsymbol{y}$ on the inputs $\boldsymbol{x}$ is non-deterministic. Therefore, although we know the data generating process, it is impossible to predict the exact value of $\boldsymbol{y}$ for a given $\boldsymbol{x}$. This type of uncertainty about the targets $\boldsymbol{y}$ is called *aleatoric uncertainty*. It is characterized by the property that it cannot be reduced, e.g., by collecting more data or by extending the hypothesis space $\mathcal{Z}$, because it is an intrinsic property of the data generating process.

In practice, we do not know $p^{\mathcal{D}}(\boldsymbol{y} \mid \boldsymbol{x})$ and want to infer a model of it from training data $\mathcal{D}^c$ drawn from this distribution. This model inference process reveals two additional sources of uncertainty. To uncover the first type of uncertainty, consider the limit $N^c \to \infty$, in which Bayesian inference yields the hypothesis $\bar{\boldsymbol{z}}$ with certainty, so that the point estimate Eq. (2.15) becomes valid. Recall from Sec. 2.1.3 that for a misspecified model the limiting hypothesis $\bar{\boldsymbol{z}}$ is erroneous in the sense that $p(\boldsymbol{y} \mid \boldsymbol{x}, \bar{\boldsymbol{z}}) \neq p^{\mathcal{D}}(\boldsymbol{y} \mid \boldsymbol{x})$. This discrepancy is due to *model uncertainty*, i.e., our uncertainty about the correct choice of the hypothesis space $\mathcal{Z}$ [Hül21]. In the limit $N^c \to \infty$, Bayesian predictions for $\boldsymbol{y}$ at $\boldsymbol{x}$ are given in terms of the plugin approximation $p(\boldsymbol{y} \mid \boldsymbol{x}, \bar{\boldsymbol{z}})$ Eq. (2.16), so that the predictive uncertainty is described as being entirely aleatoric in nature. In the absence of model uncertainty, we can choose a well-specified model with $p(\boldsymbol{y} \mid \boldsymbol{x}, \bar{\boldsymbol{z}}) = p^{\mathcal{D}}(\boldsymbol{y} \mid \boldsymbol{x})$, and the predictive uncertainty correctly captures the underlying aleatoric uncertainty. In the misspecified case, however, the plugin approximation does not recover the ground truth data generating process, and Bayesian predictions incorrectly describe the remaining uncertainty as purely aleatoric, although it is in fact a combination of aleatoric and model uncertainty (Fig. 2.3(b)).

The second type of uncertainty is *approximation uncertainty*, which we define following Hüllermeier et al. [Hül21] as the uncertainty about the correct hypothesis due to the finite amount of training data ($N^c < \infty$). In Bayesian modeling, this type of uncertainty is quantified by the posterior distribution $p(\boldsymbol{z} \mid \mathcal{D}^c)$, and translated into predictive uncertainty by means of the predictive distribution Eq. (2.8). This way of quantifying approximation uncertainty is the key distinguishing feature of a Bayesian modeling approach compared to the use of point estimates such as $\hat{\boldsymbol{z}}$ Eq. (2.14). For small training data sets, such point estimates are necessarily inaccurate and the resulting predictions overly confident because many hypotheses are compatible with the training data. In fact, in this regime, the posterior distribution can have rich structure, which explains why Bayesian predictions are often superior [Bis94, Gar18a, Wil20, Pap24].

Both approximation and model uncertainty are characterized by the property that they are due to uncertainty about the perfect model, i.e., about the model that recovers the ground truth data generating process $p^{\mathcal{D}}(\boldsymbol{y} \mid \boldsymbol{x})$. Therefore, they can — at least in principle — be reduced. In fact, approximation uncertainty decreases as more training data are observed, and model uncertainty can be reduced by extending the hypothesis space $\mathcal{Z}$ in a way that makes the model well-specified. We subsume these types of reducible uncertainties under the notion of *epistemic uncertainty*.

2.1.5 Model Parameterization

Having outlined the general approach of Bayesian modeling, we will now discuss several architectural and computational aspects in more detail. First, we consider the choice of the hypothesis space $\mathcal{Z}$, i.e., the parameterization of the model $\{p(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{z}) \mid \boldsymbol{z} \in \mathcal{Z}\}$ and of the prior distribution $p(\boldsymbol{z})$.

In Sec. 2.1.3, we established that the posterior distribution asymptotically contracts around a hypothesis $\bar{\boldsymbol{z}}$. While we showed that this is true regardless of the concrete choice of the prior distribution $p(\boldsymbol{z})$, we did not yet investigate how $p(\boldsymbol{z})$ affects the efficiency of this contraction in terms of the amount of training data required to arrive at a good estimate of $\bar{\boldsymbol{z}}$. While formal convergence rates can be established [Kle12], we restrict our attention to a view put forward by MacKay [Mac03] and Wilson et al. [Wil20], which emphasizes the marginal likelihood Eq. (2.11) as a useful quantity to reason about this problem. First, recall from Sec. 2.1.2 that the marginal likelihood Eq. (2.11) quantifies how much probability a Bayesian model assigns to a given dataset by marginalization with respect to the prior distribution $p(\boldsymbol{z})$. The marginal likelihood thus allows a quantitative comparison of different model choices, in contrast to $p(\boldsymbol{z})$, which has no meaning independent of the model parameterization.

We define the *support* of a Bayesian model as the set of datasets that have a non-vanishing marginal likelihood [Wil20]. Note that a well-specified model with $p(\bar{\boldsymbol{z}}) > 0$ is necessarily supported at the training dataset $\mathcal{D}^c$. Furthermore, we define the strength of a model's *inductive biases* towards a given dataset as the marginal likelihood of that dataset [Wil20]. Intuitively, a Bayesian model with stronger inductive biases towards $\mathcal{D}^c$ will converge to the limiting hypothesis $\bar{\boldsymbol{z}}$ more efficiently. For example, a simple model that is supported on only a small subset of the dataset space will necessarily put high prior probability mass on a small set of datasets and, thus, will have strong inductive biases towards a small class of problems. If the problem at hand happens to belong to this class, the posterior distribution will efficiently contract around this solution when observing the data. However, due to its truncated support, the model will be misspecified for most problems, where it will contract around an incorrect solution $\bar{\boldsymbol{z}}$ with $p(\boldsymbol{y} \mid \boldsymbol{x}, \bar{\boldsymbol{z}}) \neq p^{\mathcal{D}}(\boldsymbol{y} \mid \boldsymbol{x})$.

These considerations make it clear that we should parameterize our Bayesian model in a way that spends probability mass only datasets that are a priori realistic, implying as strong an inductive bias as possible toward those datasets. This is in contrast to a model that wastes probability mass on exotic datasets that will never be observed. In a generic single-task setting in engineering or science, where no specific prior knowledge is available about the task, we often use a *deep learning* approach [LeC15, Goo16, Wil20, Bis23], i.e., we parameterize the model as

$$p(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{z}) = \mathcal{N}\left(\boldsymbol{y} \mid \boldsymbol{\mu}_{\boldsymbol{z}}(\boldsymbol{x}), \operatorname{diag}\left(\boldsymbol{\sigma}^2_{\boldsymbol{z}}(\boldsymbol{x})\right)\right). \tag{2.17}$$

Here, $\boldsymbol{\mu}_{\boldsymbol{z}} : \mathcal{X} \to \mathbb{R}^{d_y}$ and $\boldsymbol{\sigma}^2_{\boldsymbol{z}} : \mathcal{X} \to \mathbb{R}^{d_y}_{+}$ are functions defined by a *multi-layer perceptron* (MLP), a specific form of a *neural network* (NN), with weights $\boldsymbol{z} \in \mathcal{Z} \subset \mathbb{R}^{d_z}$. This approach is popular because MLPs allow a flexible parameterization of general functions, provided that the dimensionality d_z of the weight space is chosen large enough [Hor89, Nea96]. Furthermore, by virtue of the *backpropagation algorithm* [Lei20, Lin76, Rum86, Goo16], NNs allow for an efficient, gradient-based computation [Cur44, Rob51, Rum86, Duc11, Hin12, Kin15] of the MLE estimate Eq. (2.14). For concreteness, we have defined a factorized *Gaussian model*, with $\mathcal{N}(\boldsymbol{y} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})$ denoting the Gaussian probability density function with mean $\boldsymbol{\mu} \in \mathbb{R}^{d_y}$ and covariance matrix[1] $\boldsymbol{\Sigma} \in \mathbb{R}^{d_y \times d_y}$, which is often a reasonable choice for regression problems [Bis06]. Note, however, that it may be unsuitable, e.g., when the target space $\mathcal{Y}$ is bounded, or when we consider classification problems where $\mathcal{Y}$ is a discrete set.

Choosing an appropriate prior distribution $p(\boldsymbol{z})$ for such a *Bayesian neural network* (BNN) can be intricate and remains an active field of research [For22a, Mur23]. In fact, mapping any kind of prior belief about the task onto a probability distribution over the high-dimensional weight space $\mathcal{Z}$ is inherently non-trivial [Rob07], and even choosing *non-informative priors* [Jef46, Jay68, Ber91, Kas96] often proves computationally intractable. In a single-task setting, we therefore often use a simple isotropic Gaussian prior of the form

$$p(\boldsymbol{z}) = \mathcal{N}\left(\boldsymbol{z} \mid \boldsymbol{\mu}_{\boldsymbol{z},0}, \operatorname{diag}\left(\boldsymbol{\sigma}^2_{\boldsymbol{z},0}\right)\right) \tag{2.18}$$

with $\boldsymbol{\mu}_{\boldsymbol{z},0} \in \mathbb{R}^{d_z}$ and $\boldsymbol{\sigma}^2_{\boldsymbol{z},0} \in \mathbb{R}^{d_z}_{+}$. While this choice is appealing from a computational point of view and allows a simple interpretation in terms of L2-regularization [Bis06], it can be suboptimal with respect to the properties of the resulting Bayesian inferences [For22b].

As discussed in Sec. 1, we often do not operate in a single-task setting, but have prior knowledge about the target task in the form of a meta-dataset of similar tasks. The meta-learning approach discussed below will allow us to *learn inductive biases towards a given class of tasks* from such a meta-dataset. Specifically, we will see that Bayesian meta-learning expresses these inductive biases in terms of a Bayesian posterior distribution over hypotheses, conditioned on the meta-dataset. To learn the target task, this posterior distribution then serves as a conventional prior distribution. Because it is informed by the meta-dataset, we call it a *meta-prior* to distinguish it from the conventional priors such as Eq. (2.18), which are chosen independently from data.

[1] We use the notation $\operatorname{diag}(\boldsymbol{\sigma}^2)$ with $\boldsymbol{\sigma}^2 \in \mathbb{R}^{d_y}$ to denote the $d_y \times d_y$ diagonal matrix with diagonal entries $\boldsymbol{\sigma}^2$.

2.1.6 Approximate Bayesian Inference

In the previous sections, we motivated the Bayesian approach to modeling by its ability to incorporate an estimate of approximation uncertainty into predictions, leading to improved generalization for small training datasets compared to point estimates. Approximation uncertainty is quantified by the posterior distribution $p(\boldsymbol{z} \mid \mathcal{D}^c)$ over hypotheses $\boldsymbol{z} \in \mathcal{Z}$. A drawback of this approach is that it is computationally intractable to compute $p(\boldsymbol{z} \mid \mathcal{D}^c)$ in all but the simplest cases, so that we usually have to resort to approximations. We now provide a brief introduction to approximate Bayesian inference, with a focus on developing a conceptual understanding of the difficulties of scaling inference to higher dimensions d_z. As we will see later, in meta-learning we can restrict Bayesian inference to only those latent features that differ from task to task, so that d_z can be kept comparatively low, allowing the use of sophisticated approximate inference techniques.

First, note that the computation of the posterior distribution according to Bayes' theorem Eq. (2.9) requires the marginal likelihood Eq. (2.11) as a normalizing factor. Computing the marginal likelihood involves solving an integral over the hypothesis space $\mathcal{Z}$, which turns out to be analytically intractable for reasonably complex models, such as the NN-based parameterization presented in Sec. 2.1.5. The reason is that the prior distribution is *non-conjugate* to the likelihood, which means that the resulting posterior distribution is not from the same (and usually simple) family of probability distributions as the prior [Mac03, Mur23, Bis06, Gel13]. Rather, the posterior distribution can have an exceedingly complex shape [Gar18a, Dra18, Wil20, Izm21, Bet18], rendering the computation of its normalization constant, i.e., the marginal likelihood, analytically intractable.

Moreover, in modern applications of Bayesian learning, such as BNNs, the posterior distribution $p(\boldsymbol{z} \mid \mathcal{D}^c)$ is not the primary object of interest. In fact, as mentioned in Sec. 2.1.5, it can be difficult to extract useful information from distributions over high-dimensional hypothesis spaces [Rob07, For22a]. Rather, we are usually more interested in predictive quantities derived from the predictive distribution Eq. (2.8). However, the predictive distribution is not only given in terms of an expectation with respect to the intractable posterior distribution, but it itself requires marginalization. Since this is intractable for the same reasons discussed above, the success of Bayesian modeling depends largely on the availability of *approximate inference* techniques that yield both accurate and computationally feasible approximations of posterior expectations of the general form

$$\mathbb{E}_{p(\boldsymbol{z} \mid \mathcal{D}^c)}\left[\mathrm{g}(\boldsymbol{z})\right] = \int \mathrm{g}(\boldsymbol{z})\, p(\boldsymbol{z} \mid \mathcal{D}^c)\, \mathrm{d}\boldsymbol{z}, \tag{2.19}$$

where $\mathrm{g} : \mathcal{Z} \to \mathbb{R}$ is some real-valued function on $\mathcal{Z}$.

2.1.6.1 Monte Carlo Estimator

Given that we can evaluate the posterior density $p(\boldsymbol{z} \mid \mathcal{D}^c)$ up to a normalizing constant, we could try to obtain a numerical approximation of the posterior expectation Eq. (2.19) using *Quasi-Monte Carlo* methods, which approximate the marginalization integral by numerical quadrature [Pre07], i.e., using an exhaustive discretization of the hypothesis space $\mathcal{Z}$. While such approaches provide low-variance estimates, they become computationally infeasible as soon as $\mathcal{Z}$ has more than a few dimensions, due to their exponential scaling behavior in the dimensionality d_z of $\mathcal{Z}$ [Gel13, Bet18].

To escape this curse of dimensionality, we need to distribute computation non-exhaustively over the hypothesis space, an insight that lies at the heart of *Monte Carlo* (MC) approximations. Such methods aim to focus on hypotheses $\boldsymbol{z} \in \mathcal{Z}$ that have dominating contributions to the posterior expectation. Assuming that $g(\boldsymbol{z})$ does not take high values in the tails of $p(\boldsymbol{z} \mid \mathcal{D}^c)$, these hypotheses lie in the *typical set* of $p(\boldsymbol{z} \mid \mathcal{D}^c)$, which is the region of $\mathcal{Z}$ where the differential probability mass $p(\boldsymbol{z} \mid \mathcal{D}^c)\,\mathrm{d}\boldsymbol{z}$ is concentrated. Unfortunately, it can be extremely difficult to quantify the typical set in higher dimensions, since it is usually not located where the posterior density $p(\boldsymbol{z} \mid \mathcal{D}^c)$ is highest, and its volume decreases relative to the volume of $\mathcal{Z}$ as d_z increases [Bet18, Mac03, Bis06]. This observation explains why approximate Bayesian inference generally becomes more difficult in higher dimensions.

The typical set perspective motivates the *MC estimator*

$$\mathbb{E}_{p(\boldsymbol{z} \mid \mathcal{D}^c)}\left[g(\boldsymbol{z})\right] \approx \frac{1}{S} \sum_{s=1}^{S} g(\boldsymbol{z}_s), \quad \boldsymbol{z}_s \sim p(\boldsymbol{z} \mid \mathcal{D}^c), \tag{2.20}$$

which focuses computation on posterior samples $\boldsymbol{z}_s$. Since, by construction, samples from the posterior distribution concentrate in its typical set, this estimator has appealing properties: it is consistent, with its standard error decreasing as $S^{-1/2}$ (regardless of d_z), so that in many cases relatively few independent samples can be sufficient to estimate the posterior expectation with reasonable accuracy [Gel13, Bis06, Mac03, Bet18].[1]

In practice, however, the Monte Carlo approach does not resolve the difficulty of scaling posterior expectations to higher dimensions d_z because it requires exact, independent samples from the posterior distribution. Obviously, such samples are not available when the posterior distribution is intractable, so the problem just got shifted from efficiently approximating marginalization integrals to efficiently obtaining independent posterior samples, which is conceptually equivalent to locating and quantifying the typical set of the posterior distribution.

[1] Note that in the special case of marginal likelihood estimation, i.e., $g(\boldsymbol{z}) = p\left(\boldsymbol{y}^c_{1:N^c} \mid \boldsymbol{x}^c_{1:N^c}, \boldsymbol{z}\right)$ and $\boldsymbol{z}_s \sim p(\boldsymbol{z})$, this approach can be inefficient if the prior $p(\boldsymbol{z})$ puts significant probability mass away from where $p\left(\boldsymbol{y}^c_{1:N^c} \mid \boldsymbol{x}^c_{1:N^c}, \boldsymbol{z}\right)$ has high values [Nea01, Gro15].

2.1.6.2 Markov Chain Monte Carlo and Variational Inference

Let us now discuss methods for computing posterior samples to evaluate Eq. (2.20). The *Markov Chain Monte Carlo* (MCMC) family of approximate inference algorithms explores the typical set using correlated samples generated by a *Markov chain* [Met53, Has70, Gem84, Dua87, Nea93, Mac03, Bet18]. These algorithms are generally considered the gold standard because they produce exact posterior samples in the limit of infinite compute resources [Bis06]. Unfortunately, they can be difficult to scale to problems with large training datasets or to complex observation models [Izm21, Bet15].

An alternative is *variational inference* (VI) [Jor99, Mac03, Bis06], which computes a tractable *variational approximation* $\hat{q}(\boldsymbol{z})$ of the posterior distribution. This approximation is defined as the *reverse KL projection* of the posterior distribution onto a *variational family*, i.e., onto some set $\mathcal{Q}$ of probability distributions over $\mathcal{Z}$, i.e.,

$$\hat{q} \in \arg\min_{q \in \mathcal{Q}} \mathrm{KL}\left[q \,\|\, p\left(\cdot \mid \mathcal{D}^c\right)\right]. \tag{2.21}$$

This form of optimization objective produces approximations that are well suited for generative purposes, since the reverse KL projection is mode-seeking in the sense that it tends to not put probability mass in regions of $\mathcal{Z}$ where the posterior distribution has no support [Bis06, Mac03].

The optimization problem Eq. (2.21) is intractable because it requires knowledge of the posterior distribution $p(\boldsymbol{z} \mid \mathcal{D}^c)$. However, we can transform it into an equivalent, but tractable, optimization problem that only depends on the likelihood and the prior,

$$\hat{q} \in \arg\max_{q \in \mathcal{Q}} \mathrm{ELBO}\left[q; \mathcal{D}^c\right], \tag{2.22}$$

where we have introduced the *evidence lower bound* (ELBO). The ELBO is a real-valued functional on $\mathcal{Q}$ of the form

$$\mathrm{ELBO}\left[q; \mathcal{D}^c\right] = \mathbb{E}_{q(\boldsymbol{z})}\left[\sum_{n=1}^{N^c} \log p\left(\boldsymbol{y}_n^c \mid \boldsymbol{x}_n^c, \boldsymbol{z}\right) + \log \frac{p(\boldsymbol{z})}{q(\boldsymbol{z})}\right]. \tag{2.23}$$

The equivalence of Eq. (2.21) with Eq. (2.22) can be established by realizing that the marginal likelihood Eq. (2.11) enjoys a decomposition of the form

$$\log p\left(\boldsymbol{y}_{1:N^c}^c \mid \boldsymbol{x}_{1:N^c}^c\right) = \mathrm{KL}\left[q \,\|\, p\left(\cdot \mid \mathcal{D}^c\right)\right] + \mathrm{ELBO}\left[q; \mathcal{D}^c\right], \tag{2.24}$$

which holds for arbitrary $q \in \mathcal{Q}$. Since the the marginal likelihood is independent of q and the ELBO bounds it from below (by positive definiteness of the KL divergence), minimizing the KL divergence is equivalent to the maximizing the ELBO.

Variational approaches generally exhibit better scaling properties than MCMC because the variational family $\mathcal{Q}$ can be chosen in a way that allows cheap posterior sampling, and because they reformulate inference as an optimization problem amenable to efficient stochastic gradient-based optimization [Rob51, Rum86, Duc11, Kin15]. In fact, modern formulations [Hof13, Ran14, Kuc17, Kin13, Mur23] define a parametric approximation, where *variational parameters* $\boldsymbol{\phi} \in \Phi \subset \mathbb{R}^{d_\phi}$ label the elements $q_{\boldsymbol{\phi}}(\boldsymbol{z})$ of $\mathcal{Q}$. The variational optimization problem Eq. (2.22) on $\mathcal{Q}$ then reduces to a standard optimization problem on Φ of the form

$$\hat{\boldsymbol{\phi}} \in \arg\max_{\boldsymbol{\phi}\in\Phi} \mathrm{ELBO}\left(\boldsymbol{\phi}; \mathcal{D}^c\right), \tag{2.25}$$

where we have overloaded notation by using $\mathrm{ELBO}\left(\boldsymbol{\phi}; \mathcal{D}^c\right)$ to denote the real-valued function on Φ defined by $\boldsymbol{\phi} \mapsto \mathrm{ELBO}\left[q_{\boldsymbol{\phi}}; \mathcal{D}^c\right]$.

2.1.6.3 Gradient Estimation

In many applications of practical interest, such as NN-based parametric models Eq. (2.17), we can easily compute the gradient $\nabla_{\boldsymbol{z}} p\left(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{z}\right)$ using standard automatic differentiation software [Pas19, Mar15]. Furthermore, many variational families $\mathcal{Q}$ of continuous probability distributions allow a *reparameterization of the expectation* in Eq. (2.23) in terms of a distribution that does not depend on $\boldsymbol{\phi}$ [Dev96]. In such cases, we obtain an unbiased and, i.p., low-variance estimate of the gradient $\nabla_{\boldsymbol{\phi}}\mathrm{ELBO}\left(\boldsymbol{\phi}; \mathcal{D}^c\right)$ for stochastic optimization from an MC estimate (2.20) of the reparameterized expectation [Kin13, Rez14, Kin19]. To distinguish this form of gradient estimation from the more sophisticated approaches discussed below, we will refer to it as *reparameterized Euclidean gradient* estimation.

Reparameterized Euclidean gradients are cheap to compute and simple to implement, so they are readily applicable even to complex parameterizations of $\mathcal{Q}$. An important example is *amortized inference* [Ger14, Kin19, Amo23] where $\boldsymbol{\phi}$ parameterizes an *inference model* $f_{\boldsymbol{\phi}}\left(\mathcal{D}^c\right)$ that defines a mapping from the space of training datasets $\mathcal{D}^c$ to $\mathcal{Q}$. Amortized inference allows modeling the inference process as a function of the training dataset $\mathcal{D}^c$. Therefore, in a single-task setting, where the training dataset $\mathcal{D}^c$ is fixed, such an approach is not necessary. However, in the multitask settings studied later in this work, inference has to be repeated for many similar datasets $\mathcal{D}^c$, and this method can lead to significant speedups.

A drawback of the simplicity of Euclidean gradients is that they can lead to suboptimal convergence rates, because they ignore that $\mathcal{Q}$, which carries the structure of a *statistical manifold* [Ama16], has non-Euclidean geometry. Replacing the Euclidean metric on Φ with the natural *Fisher information metric* on $\mathcal{Q}$ to measure distances between probability distributions yields the *natural gradient*, which can speed up convergence considerably [Ama98, Hon08, Kha17]. Similarly, the natural gradient can be combined with an *information-geometric trust region* [Pet10, Abd15, Are20, Sch15] that controls the step size through constraints in $\mathcal{Q}$ instead of using a standard learning rate to further stabilize convergence. While natural gradients and information-geometric trust regions have been successfully applied to many types of problems [Are20, Kha23], they remain more difficult to compute and implement than Euclidean gradients.

2.1.6.4 Variational Inference Gap

The challenge of VI is to find a $q_{\hat{\boldsymbol{\phi}}}(\boldsymbol{z})$ that approximates the posterior distribution $p(\boldsymbol{z} \mid \mathcal{D}^c)$ well, so that samples from $q_{\hat{\boldsymbol{\phi}}}(\boldsymbol{z})$ yield accurate estimates of posterior expectations using Eq. (2.20). This implies that the typical sets of $q_{\hat{\boldsymbol{\phi}}}(\boldsymbol{z})$ and $p(\boldsymbol{z} \mid \mathcal{D}^c)$ should have large overlap, because then samples from $q_{\hat{\boldsymbol{\phi}}}(\boldsymbol{z})$ will concentrate in the typical set of $p(\boldsymbol{z} \mid \mathcal{D}^c)$ [Bet18]. The degree to which this has been achieved can be quantified by $\mathrm{KL}\left[q_{\hat{\boldsymbol{\phi}}} \parallel p(\cdot \mid \mathcal{D}^c)\right]$, the so-called *variational inference gap* [Cre18], which, according to Eq. (2.24), is equivalent to the *ELBO looseness* $\log p\left(\boldsymbol{y}^c_{1:N^c} \mid \boldsymbol{x}^c_{1:N^c}\right) - \mathrm{ELBO}\left[q_{\hat{\boldsymbol{\phi}}}; \mathcal{D}^c\right]$. For an arbitrarily flexible choice of the variational family $\mathcal{Q}$, $q_{\hat{\boldsymbol{\phi}}}$ will exactly match the posterior distribution, resulting in a vanishing variational inference gap and a *tight ELBO.*

However, since we have assumed that it is intractable to work with the exact posterior distribution, we must use a constrained variational family $\mathcal{Q}$. This necessarily introduces approximation inaccuracies, resulting in a *variational approximation gap* that persists even in the limit of infinite computation, a property that distinguishes VI from asymptotically exact MCMC methods. In the case of amortized inference, the accuracy of the variational approximation is further degraded by inaccuracies introduced by the inference model $f_{\boldsymbol{\phi}}(\mathcal{D}^c)$, resulting in an additional *variational amortization gap.* Both gaps add up to the variational inference gap [Cre18].

2.1.6.5 Parameterization of Variational Families

A common approach to constrain $\mathcal{Q}$ is the *mean-field approximation* [Par98, Wai08], which defines $q_{\boldsymbol{\phi}}(\boldsymbol{z})$ to decompose into independent factors as

$$q_{\boldsymbol{\phi}}(\boldsymbol{z}) = \prod_{i=1}^{d_z} q_{\boldsymbol{\phi}_i}(z_i), \tag{2.26}$$

where $\boldsymbol{z} = \left(z_1, \ldots, z_{d_z}\right) \in \mathcal{Z}$ and $\boldsymbol{\phi} = \left(\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_{d_z}\right) \in \Phi$. This approximation is popular because it keeps Φ low-dimensional compared to more structured approximations, and allows for computationally cheap and robust variational optimization, sampling, and amortization [Hin93, Gra11, Hof13, Blu15, Kin19, Ble17, Hon08, Far20].

However, the mean-field assumption is often unrealistic because it does not allow correlations and multiple modes to be captured, properties commonly exhibited by posterior distributions in Bayesian learning [Bis94, Gar18a, Wil20]. This can result in suboptimal approximations that generally underestimate the posterior variance, and, thus, lead to overconfident epistemic uncertainty estimates [Tur11, Foo20b, Bis06, Mur23]. Therefore, a wide range of research explores more expressive variational families, such as structured mean-field [Sau95] or mean-field mixtures [Jaa98, Bis97], hierarchical [Ran16, Maa16] or nonparametric variational approximations [Ger12], as well as normalizing flows [Rez15, Kin16]. Recently, highly accurate *full-covariance Gaussian mixture* (GMM) variational approximations have been achieved using natural gradients [Lin20] and information-geometric trust region step size control [Are23].

2.1.6.6 Variational Expectation Maximization

We have seen that maximizing the ELBO Eq. (2.23) with respect to the variational parameters $\boldsymbol{\phi} \in \Phi$ is equivalent to computing the reverse KL projection of the posterior distribution onto the variational family $\mathcal{Q}$. We now briefly discuss the *expectation maximization* (EM) algorithm [Dem77, Bis06], another instance of ELBO maximization that will be relevant for the type of meta-model studied later in this work. We will find that certain dimensions of the hypothesis space are determined well by the training data, so that they allow a point estimate. Let us denote the subspace of the hypothesis space for which we use a point estimate by $\Theta \subset \mathbb{R}^{d_\theta}$, and the remaining space of parameters that require distributional estimates again by $\mathcal{Z}$, and let $\boldsymbol{\theta} \in \Theta$. The marginal likelihood Eq. (2.11) and the posterior distribution Eq. (2.9) then become functions of $\boldsymbol{\theta}$, and the decomposition Eq. (2.24) reads

$$\log p\left(\boldsymbol{y}^c_{1:N^c} \mid \boldsymbol{x}^c_{1:N^c}, \boldsymbol{\theta}\right) = \mathrm{KL}\left[q_{\boldsymbol{\phi}} \parallel p\left(\cdot \mid \mathcal{D}^c, \boldsymbol{\theta}\right)\right] + \mathrm{ELBO}\left(\boldsymbol{\phi}, \boldsymbol{\theta}; \mathcal{D}^c\right). \tag{2.27}$$

Note that we have also made the dependence of the ELBO on $\boldsymbol{\theta}$ explicit in the notation.

As before, maximizing the ELBO with respect to $\boldsymbol{\phi}$ (holding $\boldsymbol{\theta}$ fixed) will minimize the KL term, i.e., improve the variational approximation $q_{\boldsymbol{\phi}}$ of the posterior distribution. Assume for the moment a fully flexible variational family $\mathcal{Q}$, so that we can compute a perfect approximation with vanishing KL term, corresponding to a tight ELBO. We then call this maximization an exact *E-step*. Then, by positive definiteness of the KL divergence, maximizing the ELBO with respect to $\boldsymbol{\theta}$ (holding $\boldsymbol{\phi}$ fixed) must necessarily also increase the marginal likelihood. This maximization is called the *M-step*. Since the marginal likelihood remains constant during the E-step and increases during the M-step we converge to a *maximum marginal likelihood estimate* (MMLE), i.e.,

$$\hat{\boldsymbol{\theta}} \in \arg\max_{\boldsymbol{\theta} \in \Theta} p\left(\boldsymbol{y}^c_{1:N^c} \mid \boldsymbol{x}^c_{1:N^c}, \boldsymbol{\theta}\right), \tag{2.28}$$

by iterating successive E- and M-steps.[1] In this way, we can break down the potentially complex problem of maximizing the marginal likelihood w.r.t $\boldsymbol{\theta}$ into a sequence of simpler E- and M-steps. The same convergence guarantee holds when we perform only partial E- and M- steps, e.g., by taking single stochastic gradient steps in Φ and Θ, respectively [Bis06].

Since we are forced to work with a constrained variational family $\mathcal{Q}$, we can only perform approximate E-steps by maximizing the ELBO within $\mathcal{Q}$. Replacing the exact E-step with such an approximate version defines a variant of EM called *variational EM* [Nea98]. Variational EM is still

[1] If the E- and M-steps converge only to a *local* maximum of the ELBO, the resulting value for $\boldsymbol{\theta}$ will also be only a local maximum of the marginal likelihood.

guaranteed to converge because both the E- and the M-steps increase the ELBO. However, this no longer implies convergence to a (local) maximum of the marginal likelihood, so variational EM can give results different from $\hat{\boldsymbol{\theta}}$. We can gain an intuitive understanding of this effect by realizing that an M-step maximizes $\mathbb{E}_{q_{\boldsymbol{\phi}}}\left[\log p\left(\boldsymbol{y}^c_{1:N^c}, \boldsymbol{z} \mid \boldsymbol{x}^c_{1:N^c}, \boldsymbol{\theta}\right)\right]$ with respect to $\boldsymbol{\theta}$, which adapts the model in a way that increases the quality of hypotheses $\boldsymbol{z}$ that are likely under the current approximation $q_{\boldsymbol{\phi}}(\boldsymbol{z})$. In this way, the approximation assumptions become self-fulfilling, biasing the optimization towards a solution that is generally different from the optimal solution $\hat{\boldsymbol{\theta}}$ [Tur11, Goo16]. Despite these shortcomings, variational EM remains a computationally efficient algorithm for finding approximate maximum marginal likelihood estimates of latent variable models [Kin19].

2.1.7 Bayesian Optimization

In this section, we give a brief overview of *Bayesian Optimization* (BO) [Sno12, Sha16, Spr16], a powerful global optimization algorithm that is widely used to optimize black-box functions that are expensive to evaluate. To reduce the number of required function evaluations, BO uses a Bayesian model to quantify the current belief about the function. Its data efficiency depends heavily on the quality of the epistemic uncertainty estimates of this model (Sec. 2.1.4), which is why BO serves as an important use case and benchmark for Bayesian models.

To derive BO, we consider an *objective function* $f : \mathcal{X} \to \mathcal{Y}$, where $\mathcal{X} \subset \mathbb{R}^{d_x}$ is a compact set and $\mathcal{Y} \subset \mathbb{R}^{d_y}$. Our goal is to find a *global optimum*

$$\boldsymbol{x}^{\text{opt}} \in \arg\max_{\boldsymbol{x}\in\mathcal{X}} f(\boldsymbol{x}). \tag{2.29}$$

BO considers the black-box setting, where we have to decide for the t-th optimization iterate $\boldsymbol{x}_t \in \mathcal{X}$ based only on the *optimization history*

$$\mathcal{H}_t = \left\{(\boldsymbol{x}_n, \boldsymbol{y}_n) \in \mathcal{X} \times \mathcal{Y} \mid n \in \{1, \dots, t-1\}\right\}, \tag{2.30}$$

where $\boldsymbol{y}_n$ denotes the noise-corrupted objective function evaluation at $\boldsymbol{x}_n$. The data generating process is thus similar to Eq. (2.2), except that the inputs $\boldsymbol{x}_n$ are not drawn randomly from a distribution $p^{\mathcal{D}}(\boldsymbol{x})$ over $\mathcal{X}$, but are chosen sequentially by the optimization algorithm.

The decision for an appropriate iterate $\boldsymbol{x}_t$ based on $\mathcal{H}_t$ constitutes an instance of the *exploration-exploitation dilemma* [Sut18]. On the one hand, we want to *exploit* what we have learned so far about f, i.e., we want to place $\boldsymbol{x}_t$ in regions where we expect high objective function values based on $\mathcal{H}_t$. On the other hand, we do not want to miss regions of $\mathcal{X}$ that might yield even better results but that we have not yet *explored.*

From our discussion in the previous sections, we know that modeling $\mathcal{H}_t$ using Bayesian regression provides a well-founded solution. In fact, a Bayesian model provides a predictive distribution $p(\boldsymbol{y}_t \mid \boldsymbol{x}_t, \mathcal{H}_t)$ Eq. (2.8) that quantifies our belief about the objective function based on the optimization history $\mathcal{H}_t$. Thus, such a *surrogate model* can be used to trade off exploration against exploitation. In BO, this is achieved by defining an *acquisition function* (AF) $\alpha_t : \mathcal{X} \to \mathbb{R}$ that depends only on the surrogate model and defines the iterate $\boldsymbol{x}_t$ through the *surrogate optimization problem*

$$\boldsymbol{x}_t \in \arg\max_{\boldsymbol{x}\in\mathcal{X}} \alpha_t\left(\boldsymbol{x}_t\right). \tag{2.31}$$

Fig. 2.2 illustrates this approach.

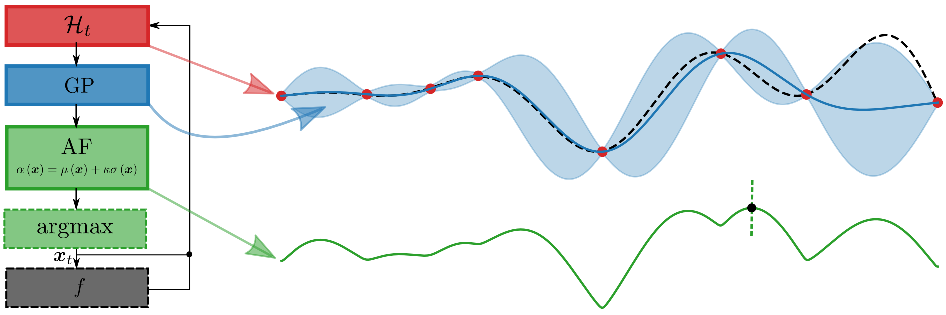

$\mathcal{H}_t$
GP
AF
$\alpha(\boldsymbol{x}) = \mu(\boldsymbol{x}) + \kappa\sigma(\boldsymbol{x})$
argmax
$\boldsymbol{x}_t$
f

Figure 2.2: Illustration of Bayesian optimization of a black-box objective function f (black, dashed). We show a Gaussian process (GP) surrogate model (blue), fitted to the optimization history $\mathcal{H}_t$ (red dots), and the UCB acquisition function (AF) α^{UCB} (green). The next optimization iterate is defined as the maximum location of the AF and is indicated by a green vertical dashed line. Note that we use the shorthand notations $\mu(\boldsymbol{x}) \equiv \mathbb{E}_{p(\boldsymbol{y}_t|\boldsymbol{x},\mathcal{H}_t)}[\boldsymbol{y}_t]$ and $\sigma^2(\boldsymbol{x}) \equiv \mathbb{V}_{p(\boldsymbol{y}_t|\boldsymbol{x},\mathcal{H}_t)}[\boldsymbol{y}_t]$ for the predictive mean and variance of the GP, respectively.

Depending on the specific properties of the optimization problem at hand, a number of surrogate models [Ras05, Hut11, Spr16] and AFs [Tho33, Kus64, Moc75, Jon98, Sri10, Hen12, Hen14] have been proposed in the literature. A prototypical instantiation of BO is given by a *Gaussian Process* (GP) [Ras05] surrogate model with the *upper confidence bound* (UCB) AF [Sri10], defined as

$$\alpha_t^{\mathrm{UCB}}\left(\boldsymbol{x}_t\right) = \mathbb{E}_{p(\boldsymbol{y}_t|\boldsymbol{x}_t,\mathcal{H}_t)}\left[\boldsymbol{y}_t\right] + \kappa_t \sqrt{\mathbb{V}_{p(\boldsymbol{y}_t|\boldsymbol{x}_t,\mathcal{H}_t)}\left[\boldsymbol{y}_t\right]}. \tag{2.32}$$

The expectation and variance are computed with respect to the predictive distribution of the surrogate model, and the corresponding terms conceptually encourage exploitation and exploration, respectively. $\kappa_t > 0$ is a scalar hyperparameter trading off these terms against each other.

A drawback of the nonparametric nature of GPs is that they can be difficult to scale to large datasets. Therefore, parametric surrogate models, as discussed in the previous sections, can be an interesting alternative [Spr16]. For such models, it is generally non-trivial to compute the required predictive expectation and variance in Eq. (2.32), but it is usually straightforward to compute approximate function samples from the underlying stochastic process (Sec. 2.1.2) from approximate posterior samples (Sec. 2.1.6). In such cases, *Thompson sampling*, where we greedily maximize these function samples, is a simple alternative AF with appealing theoretical properties [Tho33, Kan18].

While the surrogate optimization problem Eq. (2.31) can be a complex optimization problem itself [Wil18], it is typically cheaper to solve than the original problem Eq. (2.29) because no evaluations of the objective function f are required to evaluate the AF. In particular, gradients of the AF can often be easily obtained, e.g., by implementing the surrogate model and the AF in modern automatic differentiation frameworks [Mar15, Pas19, van20, Bal20], which enables efficient local optimization strategies for solving the surrogate optimization problem.

2.2 Bayesian Meta-Learning

In the previous sections, we introduced Bayesian modeling for a single-task problem, i.e., for a training dataset generated according to Eq. (2.2). We now turn our attention to settings where we have available a collection of many such datasets that share statistical structure. By applying Bayesian inference to this type of problem, we will be able to take advantage of this shared structure to speed up inference on unseen datasets.

2.2.1 Meta-Datasets and Meta-Learning

What distinguishes the problems studied in the remainder of this work from the single-task problems considered in Sec. 2.1 is that we consider inference from a *meta-dataset* $\mathcal{D}^{\text{meta}}$. A meta-dataset is defined as a dataset that is itself structured into L datasets $\mathcal{D}_\ell$ of the form Eq. (2.1), i.e.,

$$\mathcal{D}_\ell = \left\{ \left(\boldsymbol{x}_{\ell,n}, \boldsymbol{y}_{\ell,n}\right) \in \mathcal{X} \times \mathcal{Y} \mid n \in \{1, \dots, N_\ell\} \right\}. \tag{2.33}$$

The meta-dataset is defined as the disjoint union

$$\mathcal{D}^{\text{meta}} = \dot{\bigcup}_{\ell=1}^{L} \mathcal{D}_\ell. \tag{2.34}$$

Note that this construction differs from simply pooling the data (i.e., forming $\bigcup_\ell \mathcal{D}_\ell$), since we retain the additional information of how the data tuples $(\boldsymbol{x}, \boldsymbol{y})$ are grouped into distinct datasets.

We assume that the datasets $\mathcal{D}_\ell$ are generated according to Eq. (2.2), i.e., by noise-corrupted evaluations of unknown functions $f_\ell : \mathcal{X} \to \mathcal{Y}$, i.e.,

$$\boldsymbol{x}_{\ell,n} \sim p_\ell^{\mathcal{D}}(\boldsymbol{x}), \quad \boldsymbol{\varepsilon}_{\ell,n} \sim p_\ell^{\mathcal{D}}(\boldsymbol{\varepsilon} \mid \boldsymbol{x}_{\ell,n}), \quad \boldsymbol{y}_{\ell,n} = f_\ell\left(\boldsymbol{x}_{\ell,n}\right) + \boldsymbol{\varepsilon}_{\ell,n}, \tag{2.35}$$

where the f_ℓ share their domain $\mathcal{X} \subset \mathbb{R}^{d_x}$ and codomain $\mathcal{Y} \subset \mathbb{R}^{d_y}$. If this setting involves $L > 1$ such tasks, we call it a *multitask problem.* In general, the probability distributions $p_\ell^{\mathcal{D}}(\boldsymbol{x})$ over $\mathcal{X}$ and $p_\ell^{\mathcal{D}}(\boldsymbol{\varepsilon} \mid \boldsymbol{x})$ over $\mathcal{Y}$ may be task-dependent, which we indicate by the subscripts ℓ. Following Eq. (2.35), we assume that $\mathcal{D}_\ell$ is composed of independent samples from the joint data distribution

$$p_\ell^{\mathcal{D}}(\boldsymbol{x}, \boldsymbol{y}) = p_\ell^{\mathcal{D}}(\boldsymbol{y} \mid \boldsymbol{x})\, p_\ell^{\mathcal{D}}(\boldsymbol{x}), \tag{2.36}$$

where $p_\ell^{\mathcal{D}}(\boldsymbol{y} \mid \boldsymbol{x})$ is defined implicitly by Eq. (2.35) through f_ℓ and the noise process $p_\ell^{\mathcal{D}}$.

We want to perform inference from the meta-dataset $\mathcal{D}^{\text{meta}}$, i.e., we consider this dataset as the training dataset. Consequently, we call $\mathcal{D}^{\text{meta}}$ the *meta-training dataset* and the individual $\mathcal{D}_\ell$ the *training datasets*, and we aim to generalize to a *meta-test dataset*

$$\mathcal{D}^{\text{meta},*} = \dot{\bigcup}_{k=1}^{K} \mathcal{D}_k^* \tag{2.37}$$

of *test datasets*

$$\mathcal{D}_k^* = \left\{\left(\boldsymbol{x}_{k,m}^*, \boldsymbol{y}_{k,m}^*\right) \in \mathcal{X} \times \mathcal{Y} \mid m \in \{1, \dots, N_k^*\}\right\}. \tag{2.38}$$

As in the single-task case, this endeavor is only reasonable if we assume that the meta-training and meta-test datasets share statistical structure, so that observing the meta-training data provides information about the generative process of the meta-test data.

Recall our discussion in Sec. 1, where we introduced meta-learning as an approach that learns a data representation that facilitates the solution of related tasks, such as those in $\mathcal{D}^{\text{meta}}$ and $\mathcal{D}^{\text{meta},*}$. Crucially, meta-learning is derived from the multitask formulation Eq. (2.35), as opposed to transfer learning, which pre-trains the representation on a single large dataset and then fine-tunes the model to a given test task.[1] Meta-learning is in this sense similar to another approach known as *multitask learning*, but while multitask learning considers a fixed set of tasks that are learned jointly in order to benefit from a shared data representation, meta-learning aims at a representation that allows efficient generalization to unseen tasks at test time [Ben13, Zhu21, Hui21, Hos22, Bis23].

As discussed in Sec. 1, we are particularly interested in the few-shot learning scenario, where we have observed a small context dataset $\mathcal{D}_k^{c,*} \subset \mathcal{D}_k^*$ from a test dataset $\mathcal{D}_k^*$. A meta-learning approach provides an adaptation mechanism that allows efficient inference from $\mathcal{D}_k^{c,*}$ to identify properties of the test task that enable high-quality predictions on the unobserved *target dataset* $\mathcal{D}_k^{t,*} \subset \mathcal{D}_k^*$. When a test dataset $\mathcal{D}_k^*$ is completely unobserved, i.e., when $\mathcal{D}_k^{c,*} = \varnothing$ and we rely solely on $\mathcal{D}^{\text{meta}}$ to make predictions about $\mathcal{D}_k^*$, we say that we are making *zero-shot predictions.* Tab. 2.1 summarizes the datasets involved in the meta-learning problem.

[1] This single dataset can still be diverse and generated from different tasks. However, in it's vanilla formulation [Mur23, Bis23], transfer learning uses the pooled dataset $\bigcup \mathcal{D}_\ell$ and ignores the substructure we indicated in Eq. (2.34) by the disjoint union. However, recall from Sec. 1, that the boundaries between transfer and meta-learning are becoming increasingly blurred [Zhu21], especially with the advent of large foundation models [Vas17], which are technically also trained in a multitask fashion.

Table 2.1: Datasets involved in the meta-learning problem.

Name	Symbol	Purpose
training datasets	$\mathcal{D}_\ell$	all available data from the ℓ-th training task
meta-training dataset	$\mathcal{D}^{\text{meta}} = \dot{\bigcup} \mathcal{D}_\ell$	dataset of all training datasets
test datasets	$\mathcal{D}_k^*$	all available data from the k-th test task
meta-test dataset	$\mathcal{D}^{\text{meta},*} = \dot{\bigcup} \mathcal{D}_k^*$	dataset of all test datasets
(test) context datasets	$\mathcal{D}_k^{c,*} \subset \mathcal{D}_k^*$	used for adaptation to $\mathcal{D}_k^*$ during meta-testing
(test) target datasets	$\mathcal{D}_k^{t,*} \subset \mathcal{D}_k^*$	used for testing on $\mathcal{D}_k^*$ during meta-testing

2.2.2 A Bayesian Meta-Model

Let us now discuss *Bayesian Meta-Learning* (BML), i.e., Bayesian modeling for meta-datasets of the form Eq. (2.34). Throughout this text, we will be particularly interested in the zero- or few-shot settings, where the context datasets $\mathcal{D}_k^{c,*}$ are empty or small, respectively. Here, a single-task learning approach, where we ignore the meta-training dataset $\mathcal{D}^{\text{meta}}$ and make inferences based on the context datasets $\mathcal{D}_k^{c,*}$ alone, using the single-task model structure discussed in Sec. 2.1.2, will not provide satisfactory results, because predictions would be affected by large approximation uncertainty (Sec. 2.1.4).

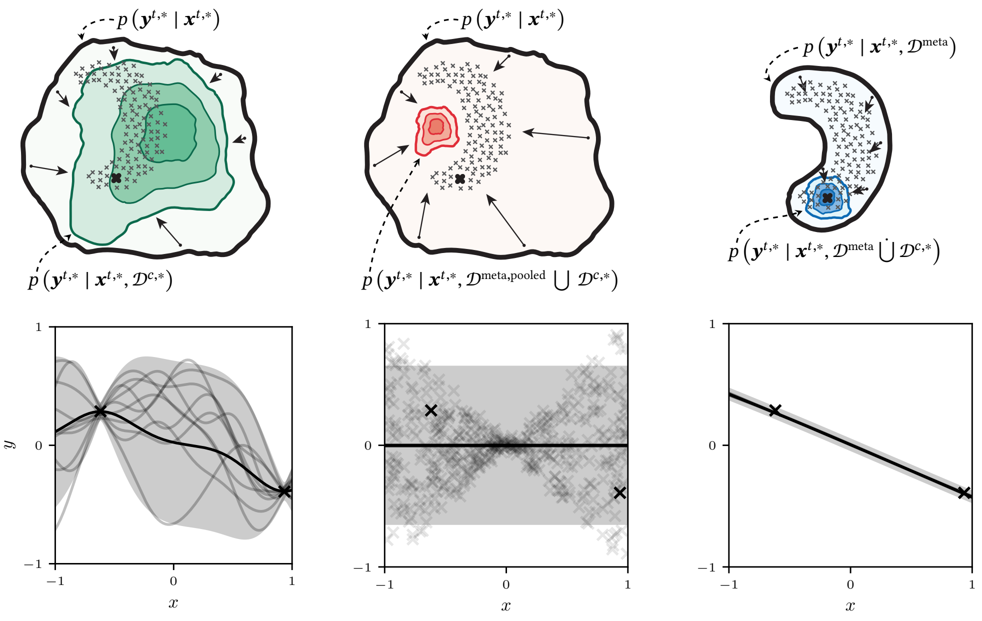
Similarly, any attempt to incorporate $\mathcal{D}^{\text{meta}}$ using such a single-task model will necessarily fail because such a model is misspecified due to its inability to represent the substructure of $\mathcal{D}^{\text{meta}}$ of individual datasets. This becomes apparent if we consider a naive *pooling* approach, where we incorporate $\mathcal{D}^{\text{meta}}$ by ignoring this substructure and perform inference on a large dataset containing all the data from $\mathcal{D}^{\text{meta}}$ and $\mathcal{D}_k^{c,*}$. Obviously, the model will then contract around an incorrect solution for $\mathcal{D}_k^{c,*}$. In fact, while the predictions will show comparatively little approximation uncertainty (because the model is informed by a large amount of data), the model misspecification will be expressed by a large model uncertainty (Sec. 2.1.4, Fig. 2.3).

These examples show that a single-task model cannot reduce epistemic uncertainty to a level sufficient for predictions of reasonable quality in a few-shot setting. To take advantage of $\mathcal{D}^{\text{meta}}$, we need to modify the model structure to obtain a well-specified model with a hypothesis space that contains a hypothesis that correctly describes the substructure of $\mathcal{D}^{\text{meta}}$. As visualized in Fig. 2.3, this will allow the model to learn inductive biases towards $\mathcal{D}^{\text{meta}}$ and efficiently contract around the correct solution for $\mathcal{D}_k^{c,*}$.

To this end, we extend the hypothesis space to $\mathcal{Z}^{\text{meta}}$, where each dataset is described by its own parameter, i.e.,

$$\boldsymbol{z}^{\text{meta}} \equiv \left(\boldsymbol{z}_1, \dots, \boldsymbol{z}_L, \boldsymbol{z}_1^*, \dots, \boldsymbol{z}_K^*\right) \in \mathcal{Z}^{\text{meta}}, \quad \mathcal{Z}^{\text{meta}} \equiv \mathcal{Z}^{L+K}. \tag{2.39}$$

Here, the parameter $\boldsymbol{z}_\ell \in \mathcal{Z}$ shall describe the ℓ-th training dataset $\mathcal{D}_\ell$, and $\boldsymbol{z}_k^* \in \mathcal{Z}$ the k-th test dataset $\mathcal{D}_k^*$. Consequently, we call these parameters *task descriptors*.



(a) Single-task approach. **(b)** Pooling approach. **(c)** Meta-learning approach.

Figure 2.3: Comparison of three Bayesian models for data drawn from linear functions ($d_x = d_y = 1$). The slopes are sampled uniformly from $[-1.0, 1.0] \subset \mathbb{R}$ to generate a meta-training dataset $\mathcal{D}^{\text{meta}}$. **Top row:** Schematic visualization of the predictive distributions in dataset space (adapted from [Wil20]). Each small cross represents a training dataset $\mathcal{D}_\ell \subset \mathcal{D}^{\text{meta}}$, corresponding to a line with a given slope. The thick cross represents a test dataset $\mathcal{D}^*$ generated from the same data distribution (with a slope of -0.4). Before observing data from the test task, predictions are made with the prior predictive distribution (light shaded area with thick black outline). The colored contours visualize the posterior predictive distribution for $\mathcal{D}^*$, given a small context dataset $\mathcal{D}^{c,*} \subset \mathcal{D}^*$. **Bottom row:** Visualization of the posterior predictive distributions in (x, y)-space. The two thick black crosses represent $\mathcal{D}^{c,*}$, the thick black line represents the predictive mean, and the shaded area represents two predictive standard deviations around the mean, indicating the predictive uncertainty. As discussed in Sec. 2.1.4, this uncertainty is a combination of aleatoric and epistemic uncertainty. To visualize the amount of epistemic uncertainty, we show noiseless function samples (light black lines), which show more variability with more epistemic uncertainty. **Panel a:** Single-task approach with a general model that ignores $\mathcal{D}^{\text{meta}}$ and uses only $\mathcal{D}^{c,*}$ for training. While the predictive distribution includes the ground truth function, it does not contract significantly around it due to the lack of training data. Therefore, the predictions are affected by a large amount of epistemic uncertainty (approximation uncertainty), so that function samples show high variability and are unlikely to represent lines. Close to the training data, however, predictive uncertainty is correctly estimated to be mainly of aleatoric kind. **Panel b:** Naive pooling approach, training the single-task model from (a) on the pooled meta-dataset $\mathcal{D}^{\text{meta,pooled}} = \bigcup_\ell \mathcal{D}_\ell$ (indicated by the light crosses in the bottom row) together with $\mathcal{D}^{c,*}$. Since this model is misspecified for $\mathcal{D}^{\text{meta}}$ (because it ignores its substructure of individual tasks), it contracts around a suboptimal solution. Model uncertainty, the cause of this model misspecification, is translated to predictive uncertainty of aleatoric kind, which represents a wrong estimate of the ground truth aleatoric uncertainty due to noise. In fact, the model computes a large predictive standard deviation, but the function samples show no variability, reflecting negligible epistemic uncertainty due to the large amount of training data. Note that all function samples are approximately equal to the predictive mean (which itself is approximately equal to the zero function due to the symmetry of the data). In effect, the predictive distribution is factorized and assigns a negligible probability density to the test data set. **Panel c:** Meta-learning approach that incorporates knowledge about the substructure of $\mathcal{D}^{\text{meta}}$, reflecting less model uncertainty. During the training phase, it uses $\mathcal{D}^{\text{meta}}$ to learn inductive biases towards linear functions in the form of a meta-prior (Sec. 2.1.5), as indicated by a prior predictive distribution that includes only the training datasets. This allows the posterior predictive distribution to efficiently contract around $\mathcal{D}^*$ using the context dataset $\mathcal{D}^{c,*}$. As a result, the predictions exhibit negligible epistemic uncertainty and correctly quantify the remaining ground truth aleatoric uncertainty.

Formally, we define the datasets to be generated independently, in the sense that the joint distribution factorizes as

$$\begin{aligned} &p\left(\boldsymbol{y}_{1:L,1:N}, \boldsymbol{y}^*_{1:K,1:N}, \boldsymbol{z}_{1:L}, \boldsymbol{z}^*_{1:K} \mid \boldsymbol{x}_{1:L,1:N}, \boldsymbol{x}^*_{1:K,1:N}\right) \\ &\equiv \prod_{\ell=1}^{L} p\left(\boldsymbol{y}_{\ell,1:N} \mid \boldsymbol{x}_{\ell,1:N}, \boldsymbol{z}_\ell\right) p\left(\boldsymbol{z}_\ell\right) \prod_{k=1}^{K} p\left(\boldsymbol{y}^*_{k,1:N} \mid \boldsymbol{x}^*_{k,1:N}, \boldsymbol{z}^*_k\right) p\left(\boldsymbol{z}^*_k\right). \end{aligned} \tag{2.40}$$

Note that we simplify notation by assuming that all datasets consist of the same number of $N_\ell = N^*_k = N$ data points. The extension to datasets of different sizes is straightforward.

Note that without any additional structure, such a model would be equivalent to $L + K$ independent single-task models trained on their respective datasets. This would not allow learning a representation of the structure shared between the tasks and using it for predictions. This motivates to further extend the parameter space by a *(task-)global parameter* $\boldsymbol{\theta} \in \Theta \subset \mathbb{R}^{d_\theta}$, conditional on which the datasets and their corresponding task descriptors are independent, i.e.,

$$\begin{aligned} &p\left(\boldsymbol{y}_{1:L,1:N}, \boldsymbol{y}^*_{1:K,1:N}, \boldsymbol{z}_{1:L}, \boldsymbol{z}^*_{1:K}, \boldsymbol{\theta} \mid \boldsymbol{x}_{1:L,1:N}, \boldsymbol{x}^*_{1:K,1:N}\right) \\ &\equiv p\left(\boldsymbol{\theta}\right) \prod_{\ell=1}^{L} p\left(\boldsymbol{y}_{\ell,1:N} \mid \boldsymbol{x}_{\ell,1:N}, \boldsymbol{z}_\ell, \boldsymbol{\theta}\right) p(\boldsymbol{z}_\ell \mid \boldsymbol{\theta}) \prod_{k=1}^{K} p\left(\boldsymbol{y}^*_{k,1:N} \mid \boldsymbol{x}^*_{k,1:N}, \boldsymbol{z}^*_k, \boldsymbol{\theta}\right) p(\boldsymbol{z}^*_k \mid \boldsymbol{\theta}). \end{aligned} \tag{2.41}$$

This means that $\boldsymbol{\theta}$ describes the structure that is common to all tasks, while the task descriptors describe the properties that vary from task to task, so that $\boldsymbol{z}_\ell$ captures the information needed in addition to $\boldsymbol{\theta}$ to describe $\mathcal{D}_\ell$ (and likewise for $\boldsymbol{z}^*_k$ and $\mathcal{D}^*_k$). Thus, for the example of linear functions from Fig. 2.3, $\boldsymbol{\theta}$ would capture the linearity of the functions and the task descriptors would capture their slopes. This model structure has been proposed for multitask learning by Heskes [Hes00] and Bakker et al. [Bak03] and is visualized in Fig. 2.4.

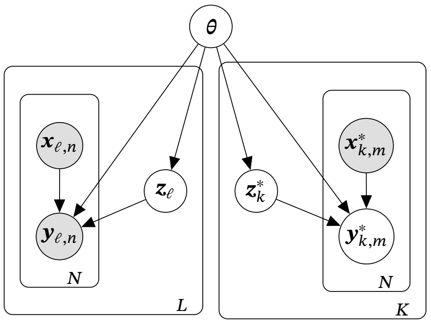


Figure 2.4: Bayesian multitask model that respects the decomposition of the data into individual tasks. The global parameter $\boldsymbol{\theta}$ describes the structure that is shared between tasks, while the task descriptors $\boldsymbol{z}_\ell$ describe the features that vary between tasks.

2.2.2.1 The Multitask Posterior Distribution

Let us first consider zero-shot predictions, i.e., predictions conditioned only on the meta-training dataset $\mathcal{D}^{\mathrm{meta}}$. Using the conditional independence assumptions induced by the multitask model Fig. 2.4, the posterior distribution over the parameters $\left(\theta, \boldsymbol{z}_{1:L}, \boldsymbol{z}^*_{1:K}\right)$ factorizes into a *global posterior* $p\left(\theta \mid \mathcal{D}^{\mathrm{meta}}\right)$, *training task posteriors* $p\left(\boldsymbol{z}_\ell \mid \mathcal{D}_\ell, \theta\right)$, and *test task priors* $p\left(\boldsymbol{z}^*_k \mid \theta\right)$ according to

$$p\left(\boldsymbol{z}_{1:L}, \boldsymbol{z}^*_{1:K}, \theta \mid \mathcal{D}^{\mathrm{meta}}\right) = p\left(\theta \mid \mathcal{D}^{\mathrm{meta}}\right) \prod_{\ell=1}^{L} p\left(\boldsymbol{z}_\ell \mid \mathcal{D}_\ell, \theta\right) \prod_{k=1}^{K} p(\boldsymbol{z}^*_k \mid \theta). \tag{2.42}$$

The resulting predictive distribution (Sec. 2.1.2) over the unobserved meta-test dataset reads

$$\begin{aligned} &p\left(\boldsymbol{y}^*_{1:K,1:N} \mid \boldsymbol{x}^*_{1:K,1:N}, \mathcal{D}^{\mathrm{meta}}\right) \\ &= \int \left(\prod_{k=1}^{K} \int \prod_{m=1}^{N} p\left(\boldsymbol{y}^*_{k,m} \mid \boldsymbol{x}^*_{k,m}, \boldsymbol{z}^*_k, \theta\right) p\left(\boldsymbol{z}^*_k \mid \theta\right) \mathrm{d}\boldsymbol{z}^*_k \right) p\left(\theta \mid \mathcal{D}^{\mathrm{meta}}\right) \mathrm{d}\theta. \end{aligned} \tag{2.43}$$

Note that the predictive distribution is independent of the training task posteriors and depends only on the global posterior and the test task priors.

According to Bayes' theorem Eq. (2.9), the global posterior reads

$$p\left(\theta \mid \mathcal{D}^{\mathrm{meta}}\right) = \frac{p\left(\boldsymbol{y}_{1:L,1:N} \mid \boldsymbol{x}_{1:L,1:N}, \theta\right) p\left(\theta\right)}{p\left(\boldsymbol{y}_{1:L,1:N} \mid \boldsymbol{x}_{1:L,1:N}\right)}. \tag{2.44}$$

The likelihood term in this equation is the marginal likelihood Eq. (2.11),

$$\begin{aligned} p\left(\boldsymbol{y}_{1:L,1:N} \mid \boldsymbol{x}_{1:L,1:N}, \theta\right) &= \int p\left(\boldsymbol{y}_{1:L,1:N} \mid \boldsymbol{x}_{1:L,1:N}, \boldsymbol{z}_{1:L}, \theta\right) p(\boldsymbol{z}_{1:L} \mid \theta)\, \mathrm{d}\boldsymbol{z}_{1:L} \\ &= \prod_{\ell=1}^{L} \int \prod_{n=1}^{N} p\left(\boldsymbol{y}_{\ell,n} \mid \boldsymbol{x}_{\ell,n}, \boldsymbol{z}_\ell, \theta\right) p(\boldsymbol{z}_\ell \mid \theta)\, \mathrm{d}\boldsymbol{z}_\ell \\ &= \prod_{\ell=1}^{L} p\left(\boldsymbol{y}_{\ell,1:N} \mid \boldsymbol{x}_{\ell,1:N}, \theta\right), \end{aligned} \tag{2.45}$$

which factorizes into *training task marginal likelihoods* $p\left(\boldsymbol{y}_{\ell,1:N} \mid \boldsymbol{x}_{\ell,1:N}, \theta\right)$. As discussed in Sec. 2.1.6, computing marginal likelihoods is intractable for all but the simplest models and will require approximation.

The training task marginal likelihoods, in turn, serve as the normalizing constants of the training task posteriors

$$p\left(\boldsymbol{z}_\ell \mid \mathcal{D}_\ell, \theta\right) = \frac{p\left(\boldsymbol{y}_{\ell,1:N} \mid \boldsymbol{x}_{\ell,1:N}, \boldsymbol{z}_\ell, \theta\right) p\left(\boldsymbol{z}_\ell \mid \theta\right)}{p\left(\boldsymbol{y}_{\ell,1:N} \mid \boldsymbol{x}_{\ell,1:N}, \theta\right)}. \tag{2.46}$$

Since the global posterior Eq. (2.44) depends on the training task marginal likelihoods, and computing them is equivalent to inferring the training task posteriors, we can say that inference of the global posterior requires inference of the training task posteriors, even though they do not appear in the zero-shot predictive distribution Eq. (2.43).

2.2.2.2 Empirical Bayes for the Global Posterior

We now discuss inference of the global posterior Eq. (2.44), which requires the training task marginal likelihoods as well as the *global marginal likelihood* $p\left(\boldsymbol{y}_{1:L,1:N} \mid \boldsymbol{x}_{1:L,1:N}\right)$, both of which are intractable. To compute an approximation, we use the MMLE $\hat{\theta}$ Eq. (2.28) for θ, i.e.,

$$p\left(\theta \mid \mathcal{D}^{\text{meta}}\right) \approx \delta\left(\theta - \hat{\theta}\right), \tag{2.47}$$

where

$$\hat{\theta} \in \arg\max_{\theta \in \Theta} p\left(\boldsymbol{y}_{1:L,1:N} \mid \boldsymbol{x}_{1:L,1:N}, \theta\right). \tag{2.48}$$

Intuitively, this is a reasonable approximation because the global posterior is informed by the full meta-dataset.[1] In Sec. 2.2.2.4 we will discuss the computation of the MMLE using variational EM as introduced in Sec. 2.1.6.6.

Under the plugin approximation Eq. (2.16), the predictive distribution Eq. (2.43) now reads

$$\begin{aligned} & p\left(\boldsymbol{y}^*_{1:K,1:N} \mid \boldsymbol{x}^*_{1:K,1:N}, \mathcal{D}^{\text{meta}}\right) \approx p\left(\boldsymbol{y}^*_{1:K,1:N} \mid \boldsymbol{x}^*_{1:K,1:N}, \hat{\theta}\right) \\ & = \prod_{k=1}^{K} \int \prod_{m=1}^{N} p\left(\boldsymbol{y}^*_{k,m} \mid \boldsymbol{x}^*_{k,m}, \boldsymbol{z}^*_k, \hat{\theta}\right) p\left(\boldsymbol{z}^*_k \mid \hat{\theta}\right) \mathrm{d}\boldsymbol{z}^*_k. \end{aligned} \tag{2.49}$$

Note that the predictive distribution now factorizes over the test datasets, and that, due to the asymptotic consensus property of Bayesian predictions (Sec. 2.1.3), the predictive distribution is independent of the choice of the global prior $p(\theta)$. Note also that the test task priors and the parametric model are now governed by the parameter $\hat{\theta}$ learned on the meta-dataset $\mathcal{D}^{\text{meta}}$. Since this is in contrast to a standard Bayesian prior that is chosen independently of the data, we refer to this approach as *empirical Bayes* [Mac92b, Bis06, Gel13]. As alluded to in Sec. 2.1.5, through this mechanism, i.e., by means of the meta-prior Eq. (2.47), we incorporate learned inductive biases towards $\mathcal{D}^{\text{meta}}$ into the predictions in BML.

2.2.2.3 Task Posteriors, Test-Task Adaptation, and Task Ambiguity

Having discussed the global posterior and zero-shot predictions, we now study the role of the task posteriors in more detail. We have already seen that the inference of the global posterior

[1] Following Sec. 2.1.3, the approximation will be correct in the limit $L \to \infty$ of infinitely many training tasks [Bak03].

Eq. (2.44) requires the inference of the training task posteriors Eq. (2.46), due to its dependence on the training task marginal likelihoods Eq. (2.45). Let us now consider the *test task posteriors.* These are required for few-shot learning, i.e., for adaptation to the test datasets. To see this, we assume the availability of context datasets $\mathcal{D}_k^{c,*} \subset \mathcal{D}_k^*$ that we want to incorporate into the predictions (Sec. 2.2.1). This is done by conditioning Eq. (2.43) on all available context data $\mathcal{D}_{1:K}^{c,*} \equiv \dot{\bigcup}_{k=1}^{K} \mathcal{D}_k^{c,*}$, and predicting the remaining target data $\mathcal{D}_{1:K}^{t,*}$, i.e.,

$$\begin{aligned} &p\left(\boldsymbol{y}_{1:K,1:N^t}^{t,*} \mid \boldsymbol{x}_{1:K,1:N^t}^{t,*}, \mathcal{D}^{\text{meta}} \,\dot{\cup}\, \mathcal{D}_{1:K}^{c,*}\right) \\ &= \int \left(\prod_{k=1}^{K} \int \prod_{m=1}^{N^t} p\left(\boldsymbol{y}_{k,m}^{t,*} \mid \boldsymbol{x}_{k,m}^{t,*}, \boldsymbol{z}_k^*, \boldsymbol{\theta}\right) p\left(\boldsymbol{z}_k^* \mid \mathcal{D}_k^{c,*}, \boldsymbol{\theta}\right) \mathrm{d}\boldsymbol{z}_k^* \right) p\left(\boldsymbol{\theta} \mid \mathcal{D}^{\text{meta}} \,\dot{\cup}\, \mathcal{D}_{1:K}^{c,*}\right) \mathrm{d}\boldsymbol{\theta}. \end{aligned} \tag{2.50}$$

Note the appearance of the test task posteriors $p\left(\boldsymbol{z}_k^* \mid \mathcal{D}_k^{c,*}, \boldsymbol{\theta}\right)$, which quantify the belief about the hypothesis $\boldsymbol{z}_k^*$ after observing the context data $\mathcal{D}_k^{c,*}$.

Note also that the global posterior is now conditioned on both the meta-training data $\mathcal{D}^{\text{meta}}$ and the context data $\mathcal{D}_{1:K}^{c,*}$, describing the refinement of the global posterior using the context data. Since we have assumed that $\mathcal{D}^{\text{meta}}$ alone is large enough to pin down $\boldsymbol{\theta}$ accurately, we can safely neglect the influence of the context datasets and make the approximation

$$p\left(\boldsymbol{\theta} \mid \mathcal{D}^{\text{meta}} \,\dot{\cup}\, \mathcal{D}_{1:K}^{c,*}\right) \approx p\left(\boldsymbol{\theta} \mid \mathcal{D}^{\text{meta}}\right) \approx \delta\left(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}\right), \tag{2.51}$$

where we used Eq. (2.47). The plugin approximation for the predictive distribution then reads

$$\begin{aligned} &p\left(\boldsymbol{y}_{1:K,1:N^t}^{t,*} \mid \boldsymbol{x}_{1:K,1:N^t}^{t,*}, \mathcal{D}^{\text{meta}} \,\dot{\cup}\, \mathcal{D}_{1:K}^{c,*}\right) \approx p\left(\boldsymbol{y}_{1:K,1:N^t}^{t,*} \mid \boldsymbol{x}_{1:K,1:N^t}^{t,*}, \mathcal{D}_{1:K}^{c,*}, \hat{\boldsymbol{\theta}}\right) \\ &= \prod_{k=1}^{K} \int \prod_{m=1}^{N^t} p\left(\boldsymbol{y}_{k,m}^{t,*} \mid \boldsymbol{x}_{k,m}^{t,*}, \boldsymbol{z}_k^*, \hat{\boldsymbol{\theta}}\right) p\left(\boldsymbol{z}_k^* \mid \mathcal{D}_k^{c,*}, \hat{\boldsymbol{\theta}}\right) \mathrm{d}\boldsymbol{z}_k^* \\ &= \prod_{k=1}^{K} p\left(\boldsymbol{y}_{k,1:N^t}^{t,*} \mid \boldsymbol{x}_{k,1:N^t}^{t,*}, \mathcal{D}_k^{c,*}, \hat{\boldsymbol{\theta}}\right). \end{aligned} \tag{2.52}$$

Note that the predictive distribution still factorizes, and that, following our discussion in Sec. 2.1.2, each of the factors $p\left(\boldsymbol{y}_{k,1:N^t}^{t,*} \mid \boldsymbol{x}_{k,1:N^t}^{t,*}, \mathcal{D}_k^{c,*}, \hat{\boldsymbol{\theta}}\right)$ is a Kolmogorov-consistent set of finite-dimensional joint distributions (for fixed $\mathcal{D}_k^{c,*}$ and $\hat{\boldsymbol{\theta}}$), which uniquely specifies a stochastic process with the joint distributions as its finite-dimensional marginals.

The approximation Eq. (2.51) allows to split BML into two phases, which can be performed sequentially. In the first phase, the so-called *meta-training phase*, we learn inductive biases towards the meta-dataset $\mathcal{D}^{\text{meta}}$ by computing the MMLE $\hat{\boldsymbol{\theta}}$. In the subsequent *meta-test phase*, we infer the test task posteriors based on a fixed $\hat{\boldsymbol{\theta}}$ in order to compute predictions with Eq. (2.52). Note that the two phases are independent in the sense that the meta-training phase can be performed without knowledge of the context datasets $\mathcal{D}_k^{c,*}$, while the meta-dataset $\mathcal{D}^{\text{meta}}$ is not required for the meta-test phase.

We have now identified the inference of the training task posteriors $p(\boldsymbol{z}_\ell \mid \mathcal{D}_\ell, \boldsymbol{\theta})$ and the test task posteriors $p\left(\boldsymbol{z}_k^* \mid \mathcal{D}_k^{c,*}, \boldsymbol{\theta}\right)$ as problems of central importance for BML. As usual, the required computations won't be analytically tractable and will require approximation. While the global posterior $p(\boldsymbol{\theta} \mid \mathcal{D}^{\text{meta}})$ is informed by the entire meta-dataset $\mathcal{D}^{\text{meta}}$ and allows a point estimate $\hat{\boldsymbol{\theta}}$ Eq. (2.51), the task posteriors are conditioned only on training datasets $\mathcal{D}_\ell$ or few-shot context datasets $\mathcal{D}_k^{c,*}$. Because these datasets are comparatively small, many different task descriptors can be compatible with the data, reflecting approximation uncertainty (Sec. 2.1.4). In the context of meta-learning, we call this concept *task ambiguity*. Consequently, the task posterior distributions can have rich structure, so we shall aim for distributional approximations to quantify approximation uncertainty and obtain robust Bayesian predictions.

The quality of the task posterior approximations will have a significant impact on the quality of the resulting predictions, both explicitly through the test task posteriors that appear in the predictive distribution Eq. (2.52), and implicitly through the training task posteriors, which influence the quality of the MMLE $\hat{\boldsymbol{\theta}}$ (Sec. 2.1.6.6). Therefore, the derivation of accurate and efficient approximation schemes for task posterior distributions in BML is a major focus of this work. The BML setting is particularly interesting in this regard because, in principle, it allows for task posterior approximations that are accurate and still relatively efficient to find. To see this, recall from Sec. 2.1.6.2 that approximate inference becomes more difficult as the dimensionality d_z of the hypothesis space $\mathcal{Z}$ increases. The interesting feature of BML is that d_z can be kept comparatively small because the task descriptors need to describe only those properties that are not shared across tasks and, thus, not already captured by the global parameter estimate $\hat{\boldsymbol{\theta}}$.

2.2.2.4 Variational EM for the Meta-Model

We now apply variational EM (Sec. 2.1.6.2) to compute approximations for the global and task posteriors introduced in the previous sections. For the global posterior, we motivated an MMLE point estimate $\hat{\boldsymbol{\theta}}$ Eq. (2.48), and for each of the task posteriors we now define a parametric variational family $\mathcal{Q}$ of approximating distributions

$$q_{\boldsymbol{\phi}_\ell}(\boldsymbol{z}_\ell) \approx p(\boldsymbol{z}_\ell \mid \mathcal{D}_\ell, \boldsymbol{\theta}), \quad q_{\boldsymbol{\phi}_k^{c,*}}\left(\boldsymbol{z}_k^*\right) \approx p\left(\boldsymbol{z}_k^* \mid \mathcal{D}_k^{c,*}, \boldsymbol{\theta}\right), \tag{2.53}$$

with variational parameters $\boldsymbol{\phi}_\ell, \boldsymbol{\phi}_k^{c,*} \in \Phi \subset \mathbb{R}^{d_\phi}$.

The decomposition Eq. (2.27) of the ℓ-th training task marginal likelihood Eq. (2.45) reads

$$\log p\left(\boldsymbol{y}_{\ell,1:N} \mid \boldsymbol{x}_{\ell,1:N}, \boldsymbol{\theta}\right) = \mathrm{KL}\left[q_{\boldsymbol{\phi}_\ell} \,\|\, p(\cdot \mid \mathcal{D}_\ell, \boldsymbol{\theta})\right] + \mathrm{ELBO}(\boldsymbol{\phi}_\ell, \boldsymbol{\theta}; \mathcal{D}_\ell), \tag{2.54}$$

with the ELBO Eq. (2.23)

$$\mathrm{ELBO}(\boldsymbol{\phi}_\ell, \boldsymbol{\theta}; \mathcal{D}_\ell) = \mathbb{E}_{q_{\boldsymbol{\phi}_\ell}(\boldsymbol{z}_\ell)}\left[\sum_{n=1}^{N} \log p\left(\boldsymbol{y}_{\ell,n} \mid \boldsymbol{x}_{\ell,n}, \boldsymbol{z}_\ell, \boldsymbol{\theta}\right) + \log \frac{p(\boldsymbol{z}_\ell \mid \boldsymbol{\theta})}{q_{\boldsymbol{\phi}_\ell}(\boldsymbol{z}_\ell)}\right]. \tag{2.55}$$

Note the appearance of the *training task priors* $p\left(\boldsymbol{z}_\ell \mid \boldsymbol{\theta}\right)$. According to Eq. (2.45), the ELBO for the full meta-dataset $\mathcal{D}^{\text{meta}}$ is then given by

$$\operatorname{ELBO}\left(\boldsymbol{\phi}_{1:L}, \boldsymbol{\theta}; \mathcal{D}^{\text{meta}}\right) = \sum_{\ell=1}^{L} \operatorname{ELBO}\left(\boldsymbol{\phi}_\ell, \boldsymbol{\theta}; \mathcal{D}_\ell\right), \tag{2.56}$$

where we denote the set of all training task variational parameters by $\boldsymbol{\phi}_{1:L} \equiv \{\boldsymbol{\phi}_1, \dots, \boldsymbol{\phi}_L\}$.

In the meta-training phase, we jointly optimize the ELBO with respect to the global parameter and the variational parameters to obtain the training task posterior approximations $q_{\hat{\boldsymbol{\phi}}_{1:L}}$ and the MMLE $\hat{\boldsymbol{\theta}}$, i.e.,

$$\left(\hat{\boldsymbol{\phi}}_{1:L}, \hat{\boldsymbol{\theta}}\right) \in \arg\max_{(\boldsymbol{\theta}, \boldsymbol{\phi}_{1:L})} \operatorname{ELBO}\left(\boldsymbol{\phi}_{1:L}, \boldsymbol{\theta}; \mathcal{D}^{\text{meta}}\right), \tag{2.57}$$

where $(\boldsymbol{\theta}, \boldsymbol{\phi}_{1:L}) \in \Theta \times \Phi^L$. Recall that the approximations $q_{\hat{\boldsymbol{\phi}}_{1:L}}$ can be discarded after the meta-training phase, as they are only needed for global posterior inference, but do not enter the predictions Eq. (2.52). Nevertheless, the quality of the approximations affects the quality of the predictions because it controls the size of the variational inference gap and, thus, the accuracy of the estimate $\hat{\boldsymbol{\theta}}$ (Sec. 2.1.6).

In the meta-test phase, we compute predictions for the test tasks $\mathcal{D}_k^{t,*}$ using Eq. (2.52). To evaluate this equation, we fix the MMLE $\hat{\boldsymbol{\theta}}$ according to Eq. (2.51) and use variational inference to approximate the test task posteriors $q_{\hat{\boldsymbol{\phi}}_k^{c,*}}\left(\boldsymbol{z}_k^*\right) \approx p\left(\boldsymbol{z}_k^* \mid \mathcal{D}_k^{c,*}, \hat{\boldsymbol{\theta}}\right)$, i.e.,

$$\hat{\boldsymbol{\phi}}_k^{c,*} \in \arg\max_{\boldsymbol{\phi}_k^{c,*} \in \Phi} \operatorname{ELBO}\left(\boldsymbol{\phi}_k^{c,*}, \hat{\boldsymbol{\theta}}; \mathcal{D}_k^{c,*}\right). \tag{2.58}$$

Note that the maximization can be performed independently for each test dataset, since $\hat{\boldsymbol{\theta}}$ is fixed.

2.2.2.5 Task Amortization

In the previous section, we discussed variational task posterior approximations of the form Eq. (2.53). Note that the computation of a task posterior approximation for a given dataset is independent of the other datasets, in the sense that we define independent variational distributions with parameters $\boldsymbol{\phi}_{1:L}$ and $\boldsymbol{\phi}_{1:K}^{c,*}$ for each of the datasets $\mathcal{D}_{1:L}$ and $\mathcal{D}_{1:K}^{c,*}$, respectively.

In meta-learning, where we assume similar datasets, we can speed up approximate inference by sharing information about the approximate inference process between datasets through *task amortization* (Sec. 2.1.6.3). To this end, we use a global variational parameter $\boldsymbol{\phi} \in \Phi$ to parameterize an inference model $f_{\boldsymbol{\phi}}\left(\mathcal{D}\right)$ that defines a mapping from the space of datasets to the variational family $\mathcal{Q}$, and which is trained to approximate inference as a function of a dataset $\mathcal{D}$. Notation-wise, we suppress the inference model and indicate task amortization by adding the dataset as an argument of the variational distributions, i.e.,

$$q_{\boldsymbol{\phi}_\ell}(\boldsymbol{z}_\ell) = q_{f_{\boldsymbol{\phi}}(\mathcal{D}_\ell)}(\boldsymbol{z}_\ell) \equiv q_{\boldsymbol{\phi}}(\boldsymbol{z}_\ell; \mathcal{D}_\ell), \quad q_{\boldsymbol{\phi}_k^{c,*}}(\boldsymbol{z}_k^*) = q_{f_{\boldsymbol{\phi}}(\mathcal{D}_k^{c,*})}(\boldsymbol{z}_k^*) \equiv q_{\boldsymbol{\phi}}(\boldsymbol{z}_k^*; \mathcal{D}_k^{c,*}). \tag{2.59}$$

The variational optimization problem Eq. (2.57) for the meta-training phase now reads

$$\left(\hat{\boldsymbol{\phi}}, \hat{\boldsymbol{\theta}}\right) \in \arg\max_{(\boldsymbol{\theta},\boldsymbol{\phi})} \mathrm{ELBO}\left(\boldsymbol{\phi}, \boldsymbol{\theta}; \mathcal{D}^{\mathrm{meta}}\right), \tag{2.60}$$

where the ELBO now depends on the two parameters $(\boldsymbol{\theta}, \boldsymbol{\phi}) \in \Theta \times \Phi$, both of which are shared between tasks. In the same spirit as in Eq. (2.51), we can assume that $\hat{\boldsymbol{\phi}}$ can be computed from $\mathcal{D}^{\mathrm{meta}}$ with sufficient accuracy, so we can fix $\hat{\boldsymbol{\phi}}$ together with $\hat{\boldsymbol{\theta}}$ after the meta-training phase. In this way, the per-task optimization Eq. (2.58) usually required to compute the test task posterior approximations during the meta-test phase can be replaced by a simple evaluation of the inference model on the context sets $\mathcal{D}_k^{c,*}$, which can significantly speed up test task posterior inference. Note, however, that task amortization introduces a variational amortization gap (Sec. 2.1.6.4) and, thus, comes at the cost of degrading the quality of the task posterior approximations, which in turn affects the quality of the MMLE estimate $\hat{\boldsymbol{\theta}}$.

2.2.3 Performance Metrics for Bayesian Meta-Learning and Meta-Optimization

In this work, we use two different performance metrics to evaluate the predictive performance of BML models and optimization algorithms. First, note that since the true predictive distribution Eq. (2.52) is intractable due to the intractability of the test task posteriors $p\left(\boldsymbol{z}_k^* \mid \mathcal{D}_k^{c,*}, \hat{\boldsymbol{\theta}}\right)$, predictions are given in terms of the *approximate predictive distribution*, which we obtain by replacing the test task posteriors with their variational approximations $q_{\hat{\boldsymbol{\phi}}_k^{c,*}}\left(\boldsymbol{z}_k^*\right)$, i.e.,

$$q\left(\boldsymbol{y}_{1:K,1:N^t}^{t,*} \mid \boldsymbol{x}_{1:K,1:N^t}^{t,*}, \mathcal{D}_{1:K}^{c,*}, \hat{\boldsymbol{\theta}}\right) \equiv \prod_{k=1}^{K} \int \prod_{m=1}^{N^t} p\left(\boldsymbol{y}_{k,m}^{t,*} \mid \boldsymbol{x}_{k,m}^{t,*}, \boldsymbol{z}_k^*, \hat{\boldsymbol{\theta}}\right) q_{\hat{\boldsymbol{\phi}}_k^{c,*}}\left(\boldsymbol{z}_k^*\right) \mathrm{d}\boldsymbol{z}_k^*. \tag{2.61}$$

More generally, if a BML algorithm uses a task posterior approximation scheme other than variational inference, we define the approximate predictive distribution accordingly, i.e., by replacing the test task posteriors with their respective approximations.

Our first metric is the *(approximate) predictive likelihood*, which we obtain by evaluating the approximate predictive distribution on the test target datasets. Since this quantity can vary over many orders of magnitude, it is usually more meaningful to report its logarithm [Mac03, Gro15].

To obtain a numerical estimate, we compute an MC estimate Eq. (2.20) of the approximate predictive likelihood, which gives

$$\begin{aligned}
&\log q\left(\boldsymbol{y}_{1:K,1:N^t}^{t,*} \mid \boldsymbol{x}_{1:K,1:N^t}^{t,*}, \mathcal{D}_{1:K}^{c,*}, \hat{\theta}\right) \\
&\gtrapprox \sum_{k=1}^{K} \log \frac{1}{S} \sum_{s=1}^{S} \prod_{m=1}^{N^t} p\left(\boldsymbol{y}_{k,m}^{t,*} \mid \boldsymbol{x}_{k,m}^{t,*}, \boldsymbol{z}_{k,s}^{*}, \hat{\theta}\right), \quad \boldsymbol{z}_{k,s}^{*} \sim q_{\hat{\phi}_k^{c,*}}\left(\boldsymbol{z}_k^{*}\right).
\end{aligned} \tag{2.62}$$

Note that the estimator on the right-hand side of this equation is not unbiased for the log approximate predictive likelihood, but merely represents a *stochastic lower bound* [Gro15, Bur16].[1] Nevertheless, the estimator is consistent [Bur16] and, thus, provides a useful metric for evaluating the predictive performance of BML models. In practice, however, the MC estimate can have high variance, so that many samples may be required to obtain an accurate estimate [Gro15, Bur16, Dub20]. As discussed in Sec. 2.1.6.1, this can be particularly severe in situations where $q_{\hat{\phi}_k^{c,*}}\left(\boldsymbol{z}_k^{*}\right)$ is a poor approximation of $p\left(\boldsymbol{z}_k^{*} \mid \mathcal{D}_k^{t,*}, \hat{\theta}\right)$, such as for simple variational families or small context sets $\mathcal{D}_k^{c,*}$.

To evaluate the performance of a global optimization algorithm such as BO (Sec. 2.1.7), we run it for a fixed budget of optimization steps and report the *simple regret* metric, defined as the difference between the best objective function evaluation observed so far and the true optimum [Sha16]. Note that downstream tasks such as optimization are also an excellent alternative evaluation strategy for BML models besides the approximate predictive likelihood [Far22]. In fact, BO is an archetypal proxy for assessing the quality of Bayesian predictions, since its success depends heavily on the quality of the surrogate model's epistemic uncertainty estimates (Sec. 2.1.7). Specifically, we can train a BML model on a meta-dataset obtained from evaluations of a set of similar objective functions. At test time, we can then use the resulting model as the surrogate for BO and report the simple regret metric.

2.2.4 The Neural Process

We now discuss the *neural process* (NP) [Gar18c], a popular family of NN-based instantiations of the Bayesian meta-model introduced in Sec. 2.2.2.

[1] Denote by M the approximate predictive likelihood and by $\hat{M}$ its MC estimate. From our discussion in Sec. 2.1.6 we know that this estimate is unbiased, i.e., $\mathbb{E}\left[\hat{M}\right] = M$. In contrast, $\log \hat{M}$ is not an unbiased estimator for the *log* approximate predictive likelihood $\log M$. In fact, by Jensen's inequality [Jen06], $\log \hat{M}$ underestimates $\log M$ in expectation, since $\mathbb{E}\left[\log \hat{M}\right] \leq \log \mathbb{E}\left[\hat{M}\right] = \log M$. Moreover, Markov's inequality [Mar84] implies that its exceedingly unlikely for $\log \hat{M}$ to significantly overestimate $\log M$ [Gro15, Bur16], which is why we call $\log \hat{M}$ a stochastic lower bound on $\log M$.

2.2.4.1 Model Architecture

In its standard formulation [Gar18c], the NP is defined by a factorized Gaussian parametric model (Sec. 2.1.5) of the form

$$p\left(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\theta}\right) = \mathcal{N}\left(\boldsymbol{y} \mid \boldsymbol{\mu}_{\boldsymbol{y},\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z}), \operatorname{diag}\left(\boldsymbol{\sigma}^2_{\boldsymbol{y},\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z})\right)\right), \tag{2.63}$$

where $\boldsymbol{\mu}_{\boldsymbol{y},\boldsymbol{\theta}} : \mathcal{X} \times \mathcal{Z} \to \mathbb{R}^{d_y}$ and $\boldsymbol{\sigma}^2_{\boldsymbol{y},\boldsymbol{\theta}} : \mathcal{X} \times \mathcal{Z} \to \mathbb{R}^{d_y}_{+}$ are functions defined by a NN with weights given by the global parameter $\boldsymbol{\theta} \in \Theta$. We call this NN the *decoder (network).* The task priors are fixed, i.e., $\boldsymbol{\theta}$-independent, isotropic Gaussian distributions of the form Eq. (2.18),

$$p\left(\boldsymbol{z}_\ell \mid \boldsymbol{\theta}\right) = \mathcal{N}\left(\boldsymbol{z}_\ell \mid \boldsymbol{\mu}_{\boldsymbol{z},0}, \operatorname{diag}\left(\boldsymbol{\sigma}^2_{\boldsymbol{z},0}\right)\right), \tag{2.64}$$

and likewise for $p\left(\boldsymbol{z}^*_k \mid \boldsymbol{\theta}\right)$. Here, $\boldsymbol{\mu}_{\boldsymbol{z},0} \in \mathbb{R}^{d_z}$ and $\boldsymbol{\sigma}^2_{\boldsymbol{z},0} \in \mathbb{R}^{d_z}_{+}$. Since the NP is trained using an empirical Bayes approach (Sec. 2.2.2.2), we do not need to specify the global prior $p\left(\boldsymbol{\theta}\right)$.

The NP uses amortized mean-field variational inference with Gaussian posterior approximations (Sec. 2.1.6) of the form

$$q_{\boldsymbol{\phi}}\left(\boldsymbol{z}_\ell; \mathcal{D}_\ell\right) = \mathcal{N}\left(\boldsymbol{z}_\ell \mid \boldsymbol{\mu}_{\boldsymbol{z},\boldsymbol{\phi}}\left(\mathcal{D}_\ell\right), \operatorname{diag}\left(\boldsymbol{\sigma}^2_{\boldsymbol{z},\boldsymbol{\phi}}\left(\mathcal{D}_\ell\right)\right)\right), \tag{2.65}$$

and likewise for $q_{\boldsymbol{\phi}}\left(\boldsymbol{z}^*_k; \mathcal{D}^{c,*}_k\right)$. The inference model $(\boldsymbol{\mu}_{\boldsymbol{z},\boldsymbol{\phi}}, \boldsymbol{\sigma}^2_{\boldsymbol{z},\boldsymbol{\phi}})$ is defined by a NN with weights $\boldsymbol{\phi} \in \Phi$, mapping from the space of datasets with elements in $\mathcal{X} \times \mathcal{Y}$ to $\mathbb{R}^{d_z}$ and $\mathbb{R}^{d_z}_{+}$, respectively. This inference model is called the *encoder (network).*

2.2.4.2 Context Aggregation

A peculiarity that distinguishes the NP architecture from similar models with an encoder-decoder structure, such as Kingma et al. [Kin13], is that the encoder network Eq. (2.65) operates on a space of *sets* with elements in $\mathcal{X} \times \mathcal{Y}$. To simplify the notation, we denote the sets $\mathcal{D}_\ell$ and $\mathcal{D}^{c,*}_k$ generically by $\mathcal{D}$ in this section. The set structure implies that the number $|\mathcal{D}|$ of input tuples $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{X} \times \mathcal{Y}$ of the encoder network Eq. (2.65) is not fixed, and that the mapping must be invariant with respect to permutations of the input tuples. These properties prevent the use of a standard NN architecture with separate input nodes for each $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{D}$. Instead, the NP uses a *(context) aggregation* mechanism, resulting in a so-called *set encoder* (Fig. 2.5).

Zaheer et al. [Zah17] and Wagstaff et al. [Wag19] characterize the general structure of such set encoders. First, each data tuple $(\boldsymbol{x}_n, \boldsymbol{y}_n) \in \mathcal{D}$ is mapped to a *latent observation* of the form

$$\boldsymbol{r}_n = \operatorname{enc}_{\boldsymbol{r},\boldsymbol{\phi}}\left(\boldsymbol{x}_n, \boldsymbol{y}_n\right) \in \mathcal{R} \subset \mathbb{R}^{d_r}, \quad n \in \{1, \dots, |\mathcal{D}|\}. \tag{2.66}$$

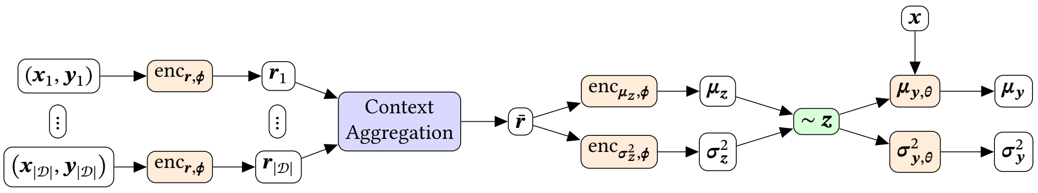
For the NP, $\operatorname{enc}_{\boldsymbol{r},\boldsymbol{\phi}} : \mathcal{X} \times \mathcal{Y} \to \mathcal{R}$ is defined by a NN whose weights we denote generically by $\boldsymbol{\phi}$, since we consider it to be part of the encoder network. Then, a permutation invariant *aggregation* *operation* is applied to the set $\{\boldsymbol{r}_1, \dots, \boldsymbol{r}_{|\mathcal{D}|}\}$ of latent observations to obtain an aggregated latent observation $\bar{\boldsymbol{r}} \in \mathcal{R}$. While the standard NP and many of its variants [Gar18c, Gar18b] use *mean aggregation* (MA), i.e.,

$$\bar{\boldsymbol{r}} \equiv \frac{1}{N} \sum_{n=1}^{|\mathcal{D}|} \boldsymbol{r}_n, \tag{2.67}$$

max aggregation [Nad20] as well as attentive [Kim19] or functional [Gor20, Foo20a] aggregation operations have also been explored in the literature. After aggregation, $\bar{\boldsymbol{r}}$ is mapped onto the parameters of the approximate task posterior distribution Eq. (2.65),

$$\boldsymbol{\mu}_{z,\boldsymbol{\phi}}(\mathcal{D}) = \mathrm{enc}_{\mu_z,\boldsymbol{\phi}}(\bar{\boldsymbol{r}}), \quad \boldsymbol{\sigma}^2_{z,\boldsymbol{\phi}}(\mathcal{D}) = \mathrm{enc}_{\sigma^2_z,\boldsymbol{\phi}}(\bar{\boldsymbol{r}}), \tag{2.68}$$

using additional NNs $\mathrm{enc}_{\mu_z,\boldsymbol{\phi}} : \mathcal{R} \to \mathbb{R}^{d_z}$ and $\mathrm{enc}_{\sigma^2_z,\boldsymbol{\phi}} : \mathcal{R} \to \mathbb{R}^{d_z}_+$.



$(\boldsymbol{x}_1, \boldsymbol{y}_1)$
$\mathrm{enc}_{\boldsymbol{r},\boldsymbol{\phi}}$
$\boldsymbol{r}_1$
$(\boldsymbol{x}_{|\mathcal{D}|}, \boldsymbol{y}_{|\mathcal{D}|})$
$\mathrm{enc}_{\boldsymbol{r},\boldsymbol{\phi}}$
$\boldsymbol{r}_{|\mathcal{D}|}$
Context Aggregation
$\bar{\boldsymbol{r}}$
$\mathrm{enc}_{\mu_z,\boldsymbol{\phi}}$
$\mathrm{enc}_{\sigma^2_z,\boldsymbol{\phi}}$
$\boldsymbol{\mu}_z$
$\boldsymbol{\sigma}^2_z$
$\sim \boldsymbol{z}$
$\boldsymbol{x}$
$\boldsymbol{\mu}_{y,\boldsymbol{\theta}}$
$\boldsymbol{\sigma}^2_{y,\boldsymbol{\theta}}$
$\boldsymbol{\mu}_y$
$\boldsymbol{\sigma}^2_y$

Figure 2.5: Computational diagram of the neural process (NP) architecture [Gar18c]. The NP is a NN-based instantiation of the Bayesian multitask model Fig. 2.4, which uses a set encoder to compute variational approximations of the task posteriors and a factorized Gaussian parametric model. See text for details.

2.2.4.3 The Train-as-you-test Paradigm

In Sec. 2.2.2.4, we determined an approximate maximizer $\hat{\boldsymbol{\theta}} \in \Theta$ of the marginal likelihood Eq. (2.45) through variational EM, i.e., by maximizing the ELBO Eq. (2.56). This ELBO decomposes into ELBOs on the task-specific marginal likelihoods, i.e.,

$$\log p\left(\boldsymbol{y}_{\ell,1:N} \mid \boldsymbol{x}_{\ell,1:N}, \boldsymbol{\theta}\right) \geq \mathrm{ELBO}\left(\boldsymbol{\phi}, \boldsymbol{\theta}; \mathcal{D}_\ell\right). \tag{2.69}$$

Furthermore, in Sec. 2.2.3, we defined the log approximate predictive likelihood Eq. (2.61), which decomposes into terms of the form

$$\log q\left(\boldsymbol{y}^{t,*}_{k,1:N^t} \mid \boldsymbol{x}^{t,*}_{k,1:N^t}, \mathcal{D}^{c,*}_k, \boldsymbol{\theta}\right), \tag{2.70}$$

as a central evaluation metric. Comparing these equations, we observe that Eq. (2.70) is informed by the context set $\mathcal{D}^{c,*}_k$, while Eq. (2.69) is based on the prior belief about the task descriptors. This reveals a mismatch between the meta-training and meta-test phases, since we optimize Eq. (2.69) in the meta-training phase, but use Eq. (2.70) to evaluate the performance of the model at meta-test time.

To resolve this mismatch, Garnelo et al. [Gar18c] modify the NP training objective and instead optimize a lower bound on the log approximate predictive likelihood with respect to the training datasets $\mathcal{D}_\ell$, i.e., on

$$\log q\left(\boldsymbol{y}^t_{\ell,1:N^t} \mid \boldsymbol{x}^t_{\ell,1:N^t}, \mathcal{D}^c_\ell, \theta\right) \equiv \log \int \prod_{n=1}^{N^t} p\left(\boldsymbol{y}^t_{\ell,n} \mid \boldsymbol{x}^t_{\ell,n}, \boldsymbol{z}_\ell, \theta\right) q_{\boldsymbol{\phi}}\left(\boldsymbol{z}_\ell; \mathcal{D}^c_\ell\right) \mathrm{d}\boldsymbol{z}_\ell. \tag{2.71}$$

Here, we have defined an *artificial context-target split* of the training datasets into *training context datasets* $\mathcal{D}^c_\ell \subset \mathcal{D}_\ell$ and *training target datasets* $\mathcal{D}^t_\ell \subset \mathcal{D}_\ell$. The corresponding ELBOs

$$\log q\left(\boldsymbol{y}^t_{\ell,1:N^t} \mid \boldsymbol{x}^t_{\ell,1:N^t}, \mathcal{D}^c_\ell, \theta\right) \geq \mathrm{ELBO}\left(\boldsymbol{\phi}, \theta; \mathcal{D}_\ell, \mathcal{D}^c_\ell\right) \tag{2.72}$$

read

$$\mathrm{ELBO}\left(\boldsymbol{\phi}, \theta; \mathcal{D}_\ell, \mathcal{D}^c_\ell\right) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}_\ell; \mathcal{D}_\ell)}\left[\sum_{n=1}^{N^t} \log p\left(\boldsymbol{y}^t_{\ell,n} \mid \boldsymbol{x}^t_{\ell,n}, \boldsymbol{z}_\ell, \theta\right) + \log \frac{q_{\boldsymbol{\phi}}\left(\boldsymbol{z}_\ell; \mathcal{D}^c_\ell\right)}{q_{\boldsymbol{\phi}}\left(\boldsymbol{z}_\ell; \mathcal{D}_\ell\right)}\right]. \tag{2.73}$$

We then use $\sum_{\ell=1}^{L} \mathrm{ELBO}\left(\boldsymbol{\phi}, \theta; \mathcal{D}_\ell, \mathcal{D}^c_\ell\right)$, possibly averaged over many different artificial context-target splits, as an alternative objective during meta-training. We say that such an approach follows the *train-as-you-test paradigm* because the meta-training procedure now mimics how the model will be used at meta-test time, as we optimize a lower bound on the evaluation metric Eq. (2.71) of interest. The standard formulation of Garnelo et al. [Gar18c] uses reparameterized Euclidean gradients (Sec. 2.1.6.3) to optimize the objective jointly with respect to $\boldsymbol{\phi}$ and θ.

There exists empirical evidence [Le18] that following the train-as-you-test paradigm can improve the predictive performance of NP models. Another alternative NP objective that follows this paradigm and often exhibits even better performance is *neural process maximum likelihood* (NPML) [Gor19, Foo20a, Dub20]. Here, we bypass a variational approach and directly maximize an MC estimate of Eq. (2.71) of the form Eq. (2.62), i.e.,

$$\log q\left(\boldsymbol{y}^t_{\ell,1:N^t} \mid \boldsymbol{x}^t_{\ell,1:N^t}, \mathcal{D}^c_\ell, \theta\right) \gtrapprox \log \frac{1}{S} \sum_{s=1}^{S} \prod_{n=1}^{N^t} p\left(\boldsymbol{y}^t_{\ell,n} \mid \boldsymbol{x}^t_{\ell,n}, \boldsymbol{z}_{\ell,s}, \theta\right), \quad \boldsymbol{z}_{\ell,s} \sim q_{\boldsymbol{\phi}}\left(\boldsymbol{z}_\ell; \mathcal{D}^c_\ell\right), \tag{2.74}$$

which can also be optimized jointly for $\boldsymbol{\phi}$ and θ using reparameterized gradients. This objective still recovers the true posterior distribution $p\left(\boldsymbol{z}_\ell \mid \mathcal{D}^c_\ell, \theta\right)$ if the approximation $q_{\boldsymbol{\phi}}\left(\boldsymbol{z}_\ell; \mathcal{D}^c_\ell\right)$ is rich enough [Gor19]. For finite model capacity, however, it encourages distributions that allow accurate predictions rather than focusing also on approximating the posterior distribution well [Dom18, Dub20].

Let us briefly discuss the relative merits of the NP objectives Eq. (2.73) and Eq. (2.74) [Dub20]. First, note that both the ELBO and NPML are lower bounds on the log approximate predictive likelihood. However, recall from Sec. 2.2.3 that NPML is a consistent estimator, so it recovers the true value in the limit of infinite samples, while the ELBO underestimates the log approximate predictive likelihood by a fixed variational inference gap (Sec. 2.1.6).[1] On the downside, NPML generally results in computationally more demanding training procedures. To see this, note that an MC estimate of the expectation in Eq. (2.73) provides an unbiased estimate of the ELBO, so single-sample estimates are usually sufficient to achieve robust convergence of stochastic gradient-based optimization to the true maximum of the ELBO [Rob51, Rum86, Duc11, Kin15]. In contrast, NPML represents a biased estimate of the log approximate predictive likelihood, so many samples may be required to achieve robust convergence. This is exacerbated by the fact that the variance of the ELBO estimator is typically lower than that of NPML for the same number of samples, in particular because for the ELBO we use samples from the approximate posterior $q_{\boldsymbol{\phi}}(\boldsymbol{z}_\ell; \mathcal{D}_\ell)$, whereas for NPML we use the less-informed distributions $q_{\boldsymbol{\phi}}(\boldsymbol{z}_\ell; \mathcal{D}_\ell^c)$ (Sec. 2.1.6 and Sec. 2.2.3).

2.2.5 Alternative Meta-Learning Approaches

In the previous sections, we studied a Bayesian multitask model (Fig. 2.4) and derived the NP as a specific NN-based instantiation. This architecture forms the basis of the meta-modeling and meta-optimization approaches developed in Sec. 3 and Sec. 4. In Sec. 5, we develop an alternative approach to meta-optimization, which is not based on a multitask model architecture, but directly meta-learns an optimization strategy. Before introducing these methods, we conclude this introductory chapter with a brief overview of alternative approaches to meta-learning. As it is an active field of research, it is beyond the scope of this work to provide a complete and consistent picture of the landscape of current meta-learning approaches, let alone a thorough evaluation of their relative merits. Therefore, we will only briefly touch on the methods that seem most related to those studied in this work, and refer the reader to recent surveys for a more comprehensive discussion [Vin17, Van19, Wan20b, Hui21, Hos22].

Recall from Sec. 2.1.3 that we motivated the Bayesian multitask model by the observation that a naive application of a single-task model in the multitask setting yields erroneous solutions under a standard Bayesian learning procedure because such a model is misspecified for the structured meta-dataset $\mathcal{D}^{\text{meta}}$ (Fig. 2.3). To alleviate this, the multitask model makes the hierarchical structure of $\mathcal{D}^{\text{meta}}$ Eq. (2.34) explicit by distinguishing between task descriptors $\boldsymbol{z}_\ell$ and a global parameter $\boldsymbol{\theta}$. This allows the application of standard approximate inference techniques such as variational EM (Sec. 2.1.6.6) for parameter estimation.

An important representative of this *model-based approach to meta-learning* [Vin17] is the NP introduced in Sec. 2.2.4 [Gar18c]. It builds on the conditional NP (CNP) [Gar18b], which defines the basic encoder-decoder architecture but does not explicitly model epistemic uncertainty about

[1] Note that this variational inference gap is not given by the KL divergences of the distributions $q_{\boldsymbol{\phi}}(\boldsymbol{z}_\ell; \mathcal{D}_\ell)$ from the posterior distribution $p(\boldsymbol{z}_\ell \mid \mathcal{D}_\ell, \boldsymbol{\theta})$ of the Bayesian model Fig. 2.4, since we are now considering a different model in which the distributions $q_{\boldsymbol{\phi}}(\boldsymbol{z}_\ell; \mathcal{D}_\ell^c)$ replace the priors $p(\boldsymbol{z}_\ell \mid \boldsymbol{\theta})$.

the task descriptor. The NP extends the CNP to an instance of the multitask model Fig. 2.4 using ideas from previous works that study several variants of the multitask model and its variational optimization objective [Edw17, Hew18, Le18]. Gordon et al. [Gor19] propose NPML Eq. (2.74) as an alternative optimization objective. A range of methods [Jha23] extend the NP architecture, e.g., by attentive computation paths to avoid underfitting [Kim19], or by bootstrapped [Lee20], graph-based [Lou19], hierarchical [Wan20a], or transformer-based [Ngu22] latent representations.

A parallel line of research aims to avoid the complexities of approximate task posterior inference (Sec. 2.1.6, [Gho20]) by performing meta-learning within the CNP framework [Gar18b]. The CNP is mathematically equivalent to an NP with a task posterior approximation of the form $p(\boldsymbol{z} \mid \mathcal{D}^c, \theta) \approx \delta\left(\boldsymbol{z} - \boldsymbol{r}_\phi(\mathcal{D}^c)\right)$, where the function $\boldsymbol{r}_\phi(\mathcal{D}^c)$ is parameterized by a set encoder as introduced in Sec. 2.2.4.2. The resulting predictive distribution Eq. (2.52) then reads

$$p\left(\boldsymbol{y}^{t,*}_{k,1:N^t} \mid \boldsymbol{x}^{t,*}_{k,1:N^t}, \mathcal{D}^{c,*}_k, \theta\right) = \prod_{m=1}^{N^t} p\left(\boldsymbol{y}^{t,*}_{k,m} \mid \boldsymbol{x}^{t,*}_{k,m}, \boldsymbol{r}_\phi\left(\mathcal{D}^{c,*}_k\right), \theta\right). \tag{2.75}$$

Since this expression does not require marginalization, the parameters ϕ and θ can be efficiently optimized during the meta-training phase by maximum likelihood on the meta-dataset $\mathcal{D}^{\text{meta}}$ (Sec. 2.1.2). At meta-test time, the maximum likelihood estimate is fixed, and the predictive distribution is obtained by a single forward pass through the architecture. Note that although the CNP does not explicitly model epistemic uncertainty about the task descriptor, it can learn during meta-training to estimate this uncertainty together with the noise variance through the output variance of the decoder. However, a severe limitation of the CNP architecture is that predictions do not model correlations between the outputs, i.e., between the elements of the set $\boldsymbol{y}^{t,*}_{k,1:N^t}$, since Eq. (2.75) factorizes over them. Consequently, the CNP does not allow to draw coherent function samples. A range of works study methods to overcome this problem, i.e., to model correlations in the output space without resorting to marginalization [Bru20, Gor20, Foo20a, Mar21, Mar22, Bru23].

We now turn our attention to *optimization-based meta-learning* methods, which can generally be characterized by a bi-level optimization scheme during the meta-training phase [Vin17, Hui21]. At the inner level, a *base learner* performs task-specific adaptation based on a context dataset, while the *outer learner* optimizes the parameters of the base learner on the meta-dataset $\mathcal{D}^{\text{meta}}$ for efficient adaptation. Notable representatives of this approach include early work by Schmidhuber [Sch87, Sch92], Bengio et al. [Ben91], and Hochreiter et al. [Hoc01], as well as hypernetworks [Ha17, Zha20] and recurrent architectures trained by gradient descent [And16, Rav17, Che17] or reinforcement learning [Li16, Li17a]. Arguably the most prominent class of optimization-based meta-learning methods is *model-agnostic meta-learning* (MAML) [Fin17], which allows any single-task model to be transformed into a meta-model without changing the model architecture. In fact, MAML makes the multitask structure of $\mathcal{D}^{\text{meta}}$ explicit in the bi-level optimization objective and is, thus, in a sense complementary to the model-based approaches studied so far, which make the multitask structure explicit in the model architecture, but leave the optimization procedure unchanged.

Formally, we consider a single-task parametric model $\left\{p\left(\boldsymbol{x} \mid \boldsymbol{y}, \boldsymbol{z}\right) \mid \boldsymbol{z} \in \mathcal{Z} \subset \mathbb{R}^{d_{\boldsymbol{z}}}\right\}$ Eq. (2.5). To obtain an optimization-based meta-learning algorithm, we need to define the base learner, i.e., how to adapt to a context dataset $\mathcal{D}_k^{c,*}$ (Sec. 2.2.1). MAML represents the base learner as K steps of gradient ascent (Sec. 2.1.5) on the log-likelihood function with respect to $\mathcal{D}_k^{c,*}$, starting from some $\theta^{\mathrm{MAML}} \in \mathcal{Z}$. Our choice of the peculiar notation θ^{MAML} for an element of $\mathcal{Z}$ will become clear below. For each dataset $\mathcal{D}_k^{c,*}$, this base learner can be described as a function $g\left(\cdot, \mathcal{D}_k^{c,*}\right) : \mathcal{Z} \to \mathcal{Z}$ mapping θ^{MAML} to the updated parameters, i.e.,

$$g\left(\theta^{\mathrm{MAML}}, \mathcal{D}_k^{c,*}\right) = \theta^{\mathrm{MAML}} + \alpha \nabla_{\boldsymbol{z}} \log p\left(\boldsymbol{y}_{k,1:N^c}^{c,*} \mid \boldsymbol{x}_{k,1:N^c}^{c,*}, \boldsymbol{z}\right)\Big|_{\theta^{\mathrm{MAML}}}, \tag{2.76}$$

where $\alpha \in \mathbb{R}$ is a learning rate parameter, and where we have set $K = 1$ to avoid cluttering the notation.

Note that the parameter $g\left(\theta^{\mathrm{MAML}}, \mathcal{D}_k^{c,*}\right) \in \mathcal{Z}$ is task-specific in the sense that it depends on the context dataset $\mathcal{D}_k^{c,*}$. Therefore, it corresponds conceptually to the task descriptor $\boldsymbol{z}_k^*$ Eq. (2.39) of the Bayesian multitask model Fig. 2.4. Moreover, $\theta^{\mathrm{MAML}} \in \mathcal{Z}$ is analogous to a point estimate for the global parameter θ of the multitask model, since adaptation starts from the same parameter θ^{MAML} for all tasks, which explains our notation. Unlike the multitask model, however, MAML's task descriptors and global parameter live in the same space $\mathcal{Z}$, i.e., in particular, they have have the same dimensionality $d_{\boldsymbol{z}}$.

From Eq. (2.76), we see that the base learner is parameterized by θ^{MAML}, which is therefore to be optimized by the outer learner on $\mathcal{D}^{\mathrm{meta}}$ during the meta-training phase. MAML fully embraces the train-as-you-test paradigm (Sec. 2.2.4.3) and defines θ^{MAML} as the maximizer of the log-likelihood function with respect to the training target datasets $\mathcal{D}_\ell^t$, evaluated at the corresponding adapted parameters $g\left(\theta^{\mathrm{MAML}}, \mathcal{D}_\ell^c\right)$, i.e,

$$\theta^{\mathrm{MAML}} \in \arg\max_{\theta \in \mathcal{Z}} \sum_{\ell=1}^{L} \log p\left(\boldsymbol{y}_{\ell,1:N^t}^t \mid \boldsymbol{x}_{\ell,1:N^t}^t, g\left(\theta, \mathcal{D}_\ell^c\right)\right). \tag{2.77}$$

Intuitively, MAML optimizes for an initialization θ^{MAML} that allows efficient adaptation through K steps of gradient descent. In its standard form, MAML's outer learner determines a local solution to this optimization problem by gradient ascent (with a learning rate parameter $\beta \in \mathbb{R}$), i.e., by iterating

$$\theta \leftarrow \theta + \beta \nabla_\theta \sum_{\ell=1}^{L} \log p\left(\boldsymbol{y}_{\ell,1:N^t}^t \mid \boldsymbol{x}_{\ell,1:N^t}^t, g\left(\theta, \mathcal{D}_\ell^c\right)\right) \tag{2.78}$$

until convergence. Note that this requires second-order gradients with respect to θ. A broad range of research aims to improve various aspects of MAML, such as performance, training stability, and computational requirements [Li17c, Ant19, Nic18, Rus18b, Raj19, Fin19].

So far, we have discussed MAML in the deterministic limit, where we do not model epistemic uncertainty (Sec. 2.1.4). In fact, we have used point estimates θ^{MAML} and $g\left(\theta^{\mathrm{MAML}}, \mathcal{D}_k^{c,*}\right)$ for both the global parameter and the task descriptors, respectively. Grant et al. [Gra18] demonstrate the equivalence of this approach to an empirical Bayes (Sec. 2.2.2.2) procedure that computes point estimates for the task descriptors in a fully hierarchical [Gel13] variant of the Bayesian multitask model (Fig. 2.4) that lacks the edge between the global parameter θ and the target variables $\boldsymbol{y}$. A number of approaches compute distributional task posterior approximations for this model within the MAML framework, i.e., by approximating task posterior inference by gradient descent from a meta-learned initialization [Gra18, Fin18, Rav19, Kim18]. Related methods perform approximate inference based on PAC-Bayesian [Ami17] or nonparametric inference techniques [Kim24].

Operating in a fully hierarchical variant of Fig. 2.4 means that the parametric model $p\left(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{z}\right)$ now depends solely on the task descriptors $\boldsymbol{z} \in \mathcal{Z}$ (instead of on both $\boldsymbol{z}$ and θ), so that the global parameter θ influences predictions only through the task priors $p\left(\boldsymbol{z} \mid \theta\right)$. For reasonably expressive parameterizations of this model, such as modern NN-based instantiations [Gra18, Fin18, Rav19, Kim18, Kim18], the dimensionality d_z of $\mathcal{Z}$, and, thus, the complexity of the resulting approximate inference problem (Sec. 2.1.6), tends to be considerably higher than for NP-like approaches (Sec. 2.2.4). To see this, note that the full weight space of the decoder now receives a distributional treatment, while for the NP this is only the case for the input space of the decoder, whose dimensionality can be chosen independently of the expressiveness of the decoder architecture. However, the fully hierarchical approach allows learning from the data the degree to which the components of the decoder's weight vector are shared across tasks, whereas the NP imposes an a-priori partitioning of the hypothesis space into fully shared decoder weights and fully task-specific task descriptors [Rav19].

3 Bayesian Context Aggregation for Neural Processes

In Sec. 2.2, we discussed the central importance of accurate task posterior approximations for solving the BML problem and introduced the standard formulation of the NP model, which computes amortized task posterior approximations using a set encoder with mean aggregation [Gar18c].

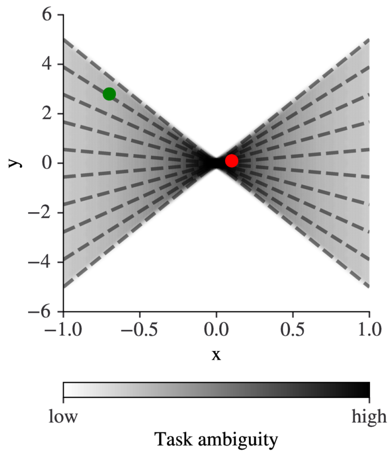


Figure 3.1: Schematic visualization of the distribution of task ambiguity in the space $\mathcal{X} \times \mathcal{Y}$ of data tuples. Similar to Fig. 2.3, the tasks are lines through the origin with varying slopes. Intuitively, the task ambiguity associated with a given context example $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{X} \times \mathcal{Y}$ decreases with its distance from the origin. A context example close to the origin (red dot) has a high task ambiguity because it could have been generated by lines with a wide range of slopes (depending on the noise level). Consequently, it provides little information about the slope (which is encoded by the task descriptor $\boldsymbol{z} \in \mathcal{Z}$), so adding it to the context set $\mathcal{D}^c$ does not significantly influence the task posterior distribution $p\left(\boldsymbol{z} \mid \mathcal{D}^c, \boldsymbol{\theta}\right)$. Conversely, the green context example carries less task ambiguity because it is compatible with a smaller range of tasks. Our Bayesian context aggregation allows such non-uniform task ambiguity distributions to be incorporated in a natural and efficient manner into the computation of amortized task posterior approximations.

In this chapter, we investigate how the context aggregation operation affects task posterior inference. While mean aggregation (like any other permutation invariant aggregation operation) in combination with the set encoder structure introduced in Sec. 2.2.4.2 can represent arbitrary functions on sets (given suitably expressive encoder NNs) [Zah17, Wag19, Gor20], we show that mean aggregation Eq. (2.67) yields a suboptimal parameterization of the task posterior approximation for NP models. Specifically, we observe that the distribution of task ambiguity (Sec. 2.2.2.3) in the space $\mathcal{X} \times \mathcal{Y}$ of data tuples is typically non-uniform, which implies that the importance of a data tuple for task posterior inference varies depending on its location (Fig. 3.1). Mean aggregation, however, assigns the same weight to each data tuple in the context set, which makes it unnecessarily difficult to learn to incorporate task ambiguity into the approximate inference scheme.

Following this observation, we develop a more efficient parameterization of context aggregation for NP set encoders, called *Bayesian context aggregation.* The central idea is to formulate context aggregation as task posterior inference by reinterpreting the latent observations $\boldsymbol{r}_n \in \mathcal{R}$ computed by the encoder NN $\mathrm{enc}_{\boldsymbol{r},\phi}$ in Eq. (2.66) as samples from a Gaussian observation model $p\left(\boldsymbol{r}_n \mid \boldsymbol{z}\right) = \mathcal{N}\left(\boldsymbol{r}_n \mid \boldsymbol{z}, \operatorname{diag}\left(\boldsymbol{\sigma}_{r_n}^2\right)\right)$. The observation model is centered at the task descriptor $\boldsymbol{z} \in \mathcal{Z}$ and, crucially, the latent observation variances $\boldsymbol{\sigma}_{r_n}^2$ are computed by additional encoder NN heads. Context aggregation then amounts to computing the posterior distribution $p\left(\boldsymbol{z} \mid \boldsymbol{r}_{1:N}\right)$, which is available in closed form through simple Gaussian conditioning [Bis06]. The latent observations $\boldsymbol{r}_n$ then appear weighted by the corresponding inverse variances $\boldsymbol{\sigma}_{r_n}^{-2}$ in the resulting parameterization of the task posterior approximation, allowing for a natural and efficient incorporation of non-uniform task ambiguity distributions into amortized approximate task posterior inference.

The remainder of this chapter has been published as [Vol21] Volpp, Michael; Flürenbrock, Fabian; Großberger, Lukas; Daniel, Christian, and Neumann, Gerhard: "Bayesian Context Aggregation for Neural Processes". In: International Conference on Learning Representations *(2021). Reprinted with permission from the authors.*

3.1 Introduction

Estimating statistical relationships between physical quantities from measured data is of central importance in all branches of science and engineering and devising powerful regression models for this purpose forms a major field of study in statistics and machine learning. When judging representative power, neural networks (NNs) are arguably the most prominent member of the regression toolbox. NNs cope well with large amounts of training data and are computationally efficient at test time. On the downside, standard NN variants do not provide uncertainty estimates over their predictions and tend to overfit on small datasets. Gaussian processes (GPs) may be viewed as complementary to NNs as they provide reliable uncertainty estimates but their cubic (quadratic) scaling with the number of context data points at training (test) time in their basic formulation affects the application on tasks with large amounts of data or on high-dimensional problems.

Recently, a lot of interest in the scientific community is drawn to combinations of aspects of NNs and GPs. Indeed, a prominent formulation of probabilistic regression is as a multi-task learning problem formalized in terms of amortized inference in conditional latent variable (CLV) models, which results in NN-based architectures which learn a distribution over target functions. Notable variants are given by the Neural Process (NP) [Gar18c] and the work of Gordon et al. [Gor19], which presents a unifying view on a range of related approaches in the language of CLV models.

Inspired by this research, we study context aggregation, a central component of such models, and propose a new, fully Bayesian, aggregation mechanism for CLV-based probabilistic regression models. To transform the information contained in the context data into a latent representation of the target function, current approaches typically employ a mean aggregator and feed the output of this aggregator into a NN to predict a distribution over global latent parameters of the function. Hence, aggregation and latent parameter inference have so far been treated as separate parts of the learning pipeline. Moreover, when using a mean aggregator, every context sample is assumed to carry the same amount of information. Yet, in practice, different input locations have different task ambiguity and, therefore, samples should be assigned different importance in the aggregation process. In contrast, our Bayesian aggregation mechanism treats context aggregation and latent parameter inference as one holistic mechanism, i.e., the aggregation directly yields the distribution over the latent parameters of the target function. Indeed, we formulate context aggregation as Bayesian inference of latent parameters using Gaussian conditioning in the latent space. Compared to existing methods, the resulting aggregator improves the handling of task ambiguity, as it can assign different variance levels to the context samples. This mechanism improves predictive performance, while it remains conceptually simple and introduces only negligible computational overhead. Moreover, our Bayesian aggregator can also be applied to deterministic model variants like the Conditional NP (CNP) [Gar18b].

In summary, our contributions are (i) a novel Bayesian Aggregation (BA) mechanism for context aggregation in NP-based models for probabilistic regression, (ii) its application to existing CLV architectures as well as to deterministic variants like the CNP, and (iii) an exhaustive experimental evaluation, demonstrating BA's superiority over traditional mean aggregation.

3.2 Related Work

Prominent approaches to probabilistic regression are Bayesian linear regression and its kernelized counterpart, the Gaussian process (GP) [Ras05]. The formal correspondence of GPs with infinite-width Bayesian NNs (BNNs) has been established in Neal [Nea96] and Williams [Wil96]. A broad range of research aims to overcome the cubic scaling behaviour of GPs with the number of context points, e.g., through sparse GP approximations [Smo01, Law02, Sne05, Qui05], by deep kernel learning [Wil16], by approximating the posterior distribution of BNNs [Mac92a, Hin93, Gal16, Lou17], or, by adaptive Bayesian linear regression, i.e., by performing inference over the last layer of a NN which introduces sparsity through linear combinations of finitely many learned basis functions [Laz10, Hin08, Sno12, Cal16]. An in a sense complementary approach aims to increase the data-efficiency of deep architectures by a fully Bayesian treatment of hierarchical latent variable models ("DeepGPs") [Dam13].

A parallel line of research studies probabilistic regression in the multi-task setting. Here, the goal is to formulate models which are data-efficient on an unseen target task by training them on data from a set of related source tasks. More general approaches of this kind employ the meta-learning framework [Sch87, Thr98, Vil05], where a model's training procedure is formulated in a way which incentivizes it to learn *how* to solve unseen tasks rapidly with only a few context examples ("learning to learn", "few-shot learning" [Fei06, Lak11]). A range of such methods trains a meta-learner to learn how to adjust the parameters of the learner's model [Ben91, Sch92], an approach which has recently been applied to few-shot image classification [Rav17], or to learning data-efficient optimization algorithms [Hoc01, Li16, And16, Che17, Per18, Vol20]. Other branches of meta-learning research aim to learn similarity metrics to determine the relevance of context samples for the target task [Koc15, Vin16, Sne17, Sun17], or explore the application of memory-augmented neural networks for meta-learning [San16]. Finn et al. [Fin17] propose model-agnostic meta-learning (MAML), a general framework for fast parameter adaptation in gradient-based learning methods.

A successful formulation of probabilistic regression as a few-shot learning problem in a multi-task setting is enabled by recent advances in the area of *probabilistic* meta-learning methods which allow a quantitative treatment of the uncertainty arising due to task ambiguity, a feature particularly relevant for few-shot learning problems. One line of work specifically studies probabilistic extensions of MAML [Gra18, Rav17, Rus18b, Fin18, Kim18]. Further important approaches are based on amortized inference in multi-task CLV models [Hes00, Bak03, Kin13, Rez14, Soh15b], which forms the basis of the Neural Statistician proposed by Edwards et al. [Edw17] and of the NP model family [Gar18c, Kim19, Lou19]. Gordon et al. [Gor19] present a unifying view on many of the aforementioned probabilistic architectures. Building on the conditional NPs (CNPs) proposed by Garnelo et al. [Gar18b], a range of NP-based architectures, such as Garnelo et al. [Gar18c] and Kim et al. [Kim19], consider combinations of deterministic and CLV model architectures. Recently, Gordon et al. [Gor20] extended CNPs to include translation equivariance in the input space, yielding state-of-the-art predictive performance.

In this paper, we also employ a formulation of probabilistic regression in terms of a multi-task CLV model. However, while in previous work the context aggregation mechanism [Zah17, Wag19] was merely viewed as a necessity to consume context sets of variable size, we take inspiration from Becker et al. [Bec19] and emphasize the fundamental connection of latent parameter inference with context aggregation and, hence, base our model on a novel Bayesian aggregation mechanism.

3.3 Preliminaries

We present the standard multi-task CLV model which forms the basis for our discussion and present traditional mean context aggregation (MA) and the variational inference (VI) likelihood approximation as employed by the NP model family [Gar18b, Kim19], as well as an alternative Monte Carlo (MC)-based approximation.

3.3.1 Problem Statement

We frame probabilistic regression as a multi-task learning problem. Let $\mathcal{F}$ denote a family of functions $f_\ell : \mathbb{R}^{d_x} \to \mathbb{R}^{d_y}$ with some form of shared statistical structure. We assume to have available data sets $\mathcal{D}_\ell \equiv \left\{\left(x_{\ell,i}, y_{\ell,i}\right)\right\}_i$ of evaluations $y_{\ell,i} \equiv f_\ell(x_{\ell,i}) + \varepsilon$ from a subset of functions ("tasks") $\{f_\ell\}_{\ell=1}^L \subset \mathcal{F}$ with additive Gaussian noise $\varepsilon \sim \mathcal{N}\left(0, \sigma_n^2\right)$. From this data, we aim to learn the posterior predictive distribution $p\left(y_\ell \mid x_\ell, \mathcal{D}_\ell^c\right)$ over a (set of) y_ℓ, given the corresponding (set of) inputs x_ℓ as well as a context set $\mathcal{D}_\ell^c \subset \mathcal{D}_\ell$.

3.3.2 The Multitask CLV Model

We formalize the multi-task learning problem in terms of a CLV model [Hes00, Gor19] as shown in Fig. 3.2.

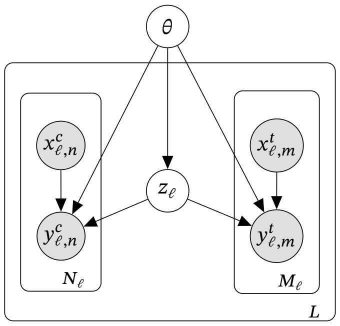


Figure 3.2: Multitask CLV model with task-specific global latent variables z_ℓ and a task-independent variable θ describing statistical structure shared between tasks.

The model employs task-specific global latent variables $z_\ell \in \mathbb{R}^{d_z}$, as well as a task-independent latent variable θ, capturing the statistical structure shared between tasks. To learn θ, we split the data into context sets $\mathcal{D}_\ell^c \equiv \{(x_{\ell,n}^c, y_{\ell,n}^c)\}_{n=1}^{N_\ell}$ and target sets $\mathcal{D}_\ell^t \equiv \{(x_{\ell,m}^t, y_{\ell,m}^t)\}_{m=1}^{M_\ell}$ and maximize the posterior predictive likelihood function

$$\prod_{\ell=1}^{L} p\left(y_{\ell,1:M_\ell}^t \mid x_{\ell,1:M_\ell}^t, \mathcal{D}_\ell^c, \theta\right) = \prod_{\ell=1}^{L} \int p\left(z_\ell \mid \mathcal{D}_\ell^c, \theta\right) \prod_{m=1}^{M_\ell} p\left(y_{\ell,m}^t \mid z_\ell, x_{\ell,m}^t, \theta\right) \mathrm{d}z_\ell \quad (3.1)$$

w.r.t. θ. In what follows, we omit task indices ℓ to avoid clutter.

3.3.3 Likelihood Approximation

Marginalizing over the task-specific latent variables z is intractable for reasonably complex models, so one has to employ some form of approximation. The NP-family of models [Gar18c, Kim19] uses an approximation of the form

$$\log p\left(y_{1:M}^t \mid x_{1:M}^t, \mathcal{D}^c, \theta\right) \gtrapprox \mathbb{E}_{q_\phi(z|\mathcal{D}^c \cup \mathcal{D}^t)}\left[\sum_{m=1}^{M} \log p\left(y_m^t \mid z, x_m^t, \theta\right) + \log \frac{q_\phi\left(z \mid \mathcal{D}^c\right)}{q_\phi\left(z \mid \mathcal{D}^c \cup \mathcal{D}^t\right)}\right]. \tag{3.2}$$

Being derived using a variational approach, this approximation utilizes an approximate posterior distribution $q_\phi\left(z \mid \mathcal{D}^c\right) \approx p\left(z \mid \mathcal{D}^c, \theta\right)$. Note, however, that it does not constitute a proper evidence lower bound for the posterior predictive likelihood since the intractable latent posterior $p\left(z \mid \mathcal{D}^c, \theta\right)$ has been replaced by $q_\phi\left(z \mid \mathcal{D}^c\right)$ in the nominator of the rightmost term [Le18]. An alternative approximation, employed for instance in Gordon et al. [Gor19], also replaces the intractable latent posterior distribution by an approximate distribution $q_\phi\left(z \mid \mathcal{D}^c\right) \approx p\left(z \mid \mathcal{D}^c, \theta\right)$ and uses a Monte-Carlo (MC) approximation of the resulting integral based on K latent samples, i.e.,

$$\log p\left(y_{1:M}^t \mid x_{1:M}^t, \mathcal{D}^c, \theta\right) \approx -\log K + \log \sum_{k=1}^{K} \prod_{m=1}^{M} p\left(y_m^t \mid z_k, x_m^t, \theta\right), \tag{3.3}$$

where $z_k \sim q_\phi\left(z \mid \mathcal{D}^c\right)$. Note that both approaches employ approximations $q_\phi\left(z \mid \mathcal{D}^c\right)$ of the latent posterior distribution $p\left(z \mid \mathcal{D}^c, \theta\right)$ and, as indicated by the notation, amortize inference in the sense that one single set of parameters ϕ is shared between all tasks. This enables efficient inference at test time, as no per-task optimization loops are required. As is standard in the literature [Gar18c, Kim19], we represent $q_\phi\left(z \mid \mathcal{D}^c\right)$ and $p\left(y_m^t \mid z, x_m^t, \theta\right)$ by NNs and refer to them as the encoder (enc, parameters ϕ) and decoder (dec, parameters θ) networks, respectively. These networks set the means and variances of factorized Gaussian distributions, i.e.,

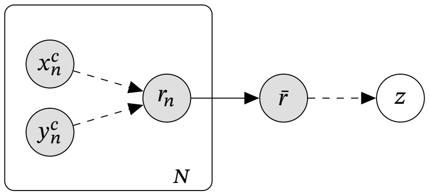
$$q_\phi\left(z \mid \mathcal{D}^c\right) = \mathcal{N}\left(z \mid \mu_z, \operatorname{diag}\left(\sigma_z^2\right)\right), \quad \mu_z = \operatorname{enc}_{\mu_z,\phi}\left(\mathcal{D}^c\right), \quad \sigma_z^2 = \operatorname{enc}_{\sigma_z^2,\phi}\left(\mathcal{D}^c\right), \tag{3.4}$$

and

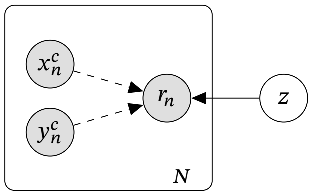
$$p\left(y_m^t \mid z, x_m^t, \theta\right) = \mathcal{N}\left(y_m^t \mid \mu_y, \operatorname{diag}\left(\sigma_y^2\right)\right), \quad \mu_y = \operatorname{dec}_{\mu_y,\theta}\left(z, x_m^t\right), \quad \sigma_y^2 = \operatorname{dec}_{\sigma_y^2,\theta}\left(z, x_m^t\right). \tag{3.5}$$

3.3.4 Context Aggregation

The latent variable z is global in the sense that it depends on the whole context set $\mathcal{D}^c$. Therefore, some form of aggregation mechanism is required to enable the encoder to consume context sets $\mathcal{D}^c$ of variable size. To represent a meaningful operation on sets, such an aggregation mechanism has to be invariant to permutations of the context data points. Zaheer et al. [Zah17] characterize possible aggregation mechanisms w.r.t. this permutation invariance condition, resulting in the structure of traditional aggregation mechanisms depicted in Fig. 3.3a.



(a) Traditional mean aggregation (MA).



(b) Our Bayesian aggregation (BA).

Figure 3.3: Comparison of aggregation mechanisms in CLV models. Dashed lines correspond to learned components of the posterior approximation $q_\phi(z \mid \mathcal{D}^c)$. BA avoids the detour via a mean-aggregated latent observation $\bar{r}$ and aggregates $\mathcal{D}^c$ directly in the statistical description of z. This allows to incorporate a quantification of the information content of each context tuple (x_n^c, y_n^c) as well as of z into the inference in a principled manner, while MA assigns the same weight to each context tuple.

Each context data tuple (x_n^c, y_n^c) is first mapped onto a latent observation

$$r_n = \mathrm{enc}_{r,\phi}(x_n^c, y_n^c) \in \mathbb{R}^{d_r}. \tag{3.6}$$

Then, a permutation-invariant operation is applied to the set $\{r_n\}_{n=1}^N$ to obtain an aggregated latent observation $\bar{r}$. One prominent choice, employed for instance in Garnelo et al. [Gar18b], Kim et al. [Kim19], and Gordon et al. [Gor19], is to take the mean, i.e.,

$$\bar{r} = \frac{1}{N}\sum_{n=1}^{N} r_n. \tag{3.7}$$

Subsequently, $\bar{r}$ is mapped onto the parameters μ_z and σ_z^2 of the approximate posterior distribution $q_\phi(z \mid \mathcal{D}^c)$ using additional encoder networks, i.e., $\mu_z = \mathrm{enc}_{\mu_z,\phi}(\bar{r})$ and $\sigma_z^2 = \mathrm{enc}_{\sigma_z^2,\phi}(\bar{r})$. Note that three encoder networks are employed here: (i) $\mathrm{enc}_{r,\phi}$ to map from the context pairs to r_n, (ii) $\mathrm{enc}_{\mu_z,\phi}$ to compute μ_z from the aggregated mean $\bar{r}$ and (iii) $\mathrm{enc}_{\sigma_z^2,\phi}$ to compute the variance σ_z^2 from $\bar{r}$. In what follows, we refer to this aggregation mechanism as mean aggregation (MA) and to the networks $\mathrm{enc}_{\mu_z,\phi}$ and $\mathrm{enc}_{\sigma_z^2,\phi}$ collectively as "$\bar{r}$-to-z-networks".

3.4 Bayesian Context Aggregation

We propose Bayesian Aggregation (BA), a novel context data aggregation technique for CLV models which avoids the detour via an aggregated latent observation $\bar{r}$ and directly treats the object of interest, namely the latent variable z, as the aggregated quantity. This reflects a central observation for CLV models with global latent variables: *context data aggregation and hidden parameter inference are fundamentally the same mechanism.* Our key insight is to define a probabilistic observation model $p(r \mid z)$ for r which depends on z. Given a new latent observation $r_n = \mathrm{enc}_{r,\phi}(x_n^c, y_n^c)$, we can update $p(z)$ by computing the posterior $p(z \mid r_n) = p(r_n \mid z)p(z)/p(r_n)$. Hence, by formulating context data aggregation as a Bayesian inference problem, we aggregate the information contained in $\mathcal{D}^c$ directly into the statistical description of z based on first principles.

3.4.1 Bayesian Context Aggregation via Gaussian Conditioning

BA can easily be implemented using a factorized Gaussian observation model of the form

$$p\left(r_n \mid z\right) = \mathcal{N}\left(r_n \mid z, \operatorname{diag}(\sigma^2_{r_n})\right), \quad r_n = \operatorname{enc}_{r,\phi}\left(x^c_n, y^c_n\right), \quad \sigma^2_{r_n} = \operatorname{enc}_{\sigma^2_r,\phi}\left(x^c_n, y^c_n\right). \tag{3.8}$$

Note that, in contrast to standard variational auto-encoders (VAEs) [Kin13], we do not learn the mean and variance of a Gaussian distribution, but we learn the latent observation r_n (which can be considered as a sample of $p(z)$) together with the variance $\sigma^2_{r_n}$ of this observation. This architecture allows the application of Gaussian conditioning while this is difficult for VAEs. Indeed, we impose a factorized Gaussian prior

$$p_0\left(z\right) \equiv \mathcal{N}\left(z \mid \mu_{z,0}, \operatorname{diag}\left(\sigma^2_{z,0}\right)\right), \tag{3.9}$$

and arrive at a Gaussian aggregation model which allows to derive the parameters of the posterior distribution $q_\phi\left(z \mid \mathcal{D}^c\right)$ in closed form[1] (cf. App. A.1):

$$\sigma^2_z = \left[\left(\sigma^2_{z,0}\right)^\ominus + \sum_{n=1}^{N}\left(\sigma^2_{r_n}\right)^\ominus\right]^\ominus, \quad \mu_z = \mu_{z,0} + \sigma^2_z \odot \sum_{n=1}^{N}\left(r_n - \mu_{z,0}\right) \oslash \left(\sigma^2_{r_n}\right). \tag{3.10}$$

Here $^\ominus$, $\odot$ and $\oslash$ denote element-wise inversion, product, and division, respectively. These equations naturally lend themselves to efficient incremental updates as new context data (x^c_n, y^c_n) arrives by using the current posterior parameters $\mu_{z,\text{old}}$ and $\sigma^2_{z,\text{old}}$ in place of the prior parameters, i.e.,

$$\sigma^2_{z,\text{new}} = \left[\left(\sigma^2_{z,\text{old}}\right)^\ominus + \left(\sigma^2_{r_n}\right)^\ominus\right]^\ominus, \quad \mu_z = \mu_{z,\text{old}} + \sigma^2_{z,\text{new}} \odot \left(r_n - \mu_{z,\text{old}}\right) \oslash \left(\sigma^2_{r_n}\right). \tag{3.11}$$

BA employs two encoder networks, $\operatorname{enc}_{r,\phi}$ and $\operatorname{enc}_{\sigma^2_r,\phi}$, mapping context tuples to latent observations and their variances, respectively. In contrast to MA, it does not require $\bar{r}$-to-z-networks, because the set $\{r_n\}_{n=1}^N$ is aggregated directly into the statistical description of z by means of Eq. (3.10), cf. Fig. 3.3b. Note that our factorization assumptions avoid the expensive matrix inversions that typically occur in Gaussian conditioning and which are difficult to backpropagate. Using factorized distributions renders BA cheap to evaluate with only marginal computational

[1] Note that an extended observation model of the form $p\left(r_n \mid z\right) = \mathcal{N}\left(r_n \mid z + \mu_{r_n}, \operatorname{diag}(\sigma^2_{r_n})\right)$, with μ_{r_n} given by a third encoder output, does not lead to a more expressive aggregation mechanism. Indeed, the resulting posterior variances would stay unchanged and the posterior mean would read $\mu_z = \mu_{z,0} + \sigma^2_z \odot \sum_{n=1}^{N}\left(r_n - \mu_{r_n} - \mu_{z,0}\right) \oslash \left(\sigma^2_{r_n}\right)$. Therefore, we would just subtract two distinct encoder outputs computed from the same inputs, resulting in exactly the same expressivity, which is why we set $\mu_{r_n} \equiv 0$.

overhead in comparison to MA. Furthermore, we can easily backpropagate through BA to compute gradients to optimize the parameters of the encoder and decoder networks. As the latent space z is shaped by the encoder network, the factorization assumptions are valid because the network will find a space where these assumptions work well. Note further that BA represents a permutation-invariant operation on $\mathcal{D}^c$.

3.4.2 Discussion

BA includes MA as a special case. Indeed, Eq. (3.10) reduces to the mean-aggregated latent observation Eq. (3.7) if we impose a non-informative prior and uniform observation variances $\sigma_{r_n}^2 \equiv 1$.[1] This observation sheds light on the benefits of a Bayesian treatment of aggregation. MA assigns the same weight $1/N$ to each latent observation r_n, independent of the amount of information contained in the corresponding context data tuple (x_n^c, y_n^c), as well as independent of the uncertainty about the current estimation of z. Bayesian aggregation remedies both of these limitations: the influence of r_n on the parameters $\mu_{z,\mathrm{old}}$ and $\sigma_{z,\mathrm{old}}^2$ describing the current aggregated state is determined by the relative magnitude of the observation variance $\sigma_{r_n}^2$ and the latent variance $\sigma_{z,\mathrm{old}}^2$, cf. Eq. (3.11). This emphasizes the central role of the learned observation variances $\sigma_{r_n}^2$: they allow to quantify the amount of information contained in each latent observation r_n. BA can therefore handle task ambiguity more efficiently than MA, as the architecture can learn to assign little weight (by predicting high observation variances $\sigma_{r_n}^2$) to context points (x_n^c, y_n^c) located in areas with high task ambiguity, i.e., to points which could have been generated by many of the functions in $\mathcal{F}$. Conversely, in areas with little task ambiguity, i.e., if (x_n^c, y_n^c) contains a lot of information about the underlying function, BA can induce a strong influence on the posterior latent distribution. In contrast, MA has to find ways to propagate such information through the aggregation mechanism by encoding it in the mean-aggregated latent observation $\bar{r}$.

3.4.3 Likelihood Approximation with Bayesian Context Aggregation

We show that BA is versatile in the sense that it can replace traditional MA in various CLV-based NP architectures as proposed, e.g., in Garnelo et al. [Gar18c] and Gordon et al. [Gor19], which employ samples from the approximate latent posterior $q_\phi(z \mid \mathcal{D}^c)$ to approximate the likelihood (as discussed in Sec. 3.3), as well as in deterministic variants like the CNP [Gar18b].

[1] As motivated above, we consider $\bar{r}$ as the aggregated quantity of MA and the distribution over z, described by μ_z and σ_z^2, as the aggregated quantity of BA. Note that Eq. (3.10) does not necessarily generalize μ_z and σ_z^2 after nonlinear $\bar{r}$-to-z-networks.

3.4.3.1 Sampling-Based Likelihood Approximations

BA is naturally compatible with both the VI and MC likelihood approximations for CLV models. Indeed, BA defines a Gaussian latent distribution from which we can easily obtain samples z in order to evaluate Eq. (3.2) or Eq. (3.3) using the decoder parameterization Eq. (3.5).

3.4.3.2 Bayesian Context Aggregation for Conditional Neural Processes

BA motivates a novel, alternative, method to approximate the posterior predictive likelihood Eq. (3.1), resulting in a deterministic loss function which can be efficiently optimized for θ and ϕ in an end-to-end fashion. To this end, we employ a Gaussian approximation of the posterior predictive likelihood of the form

$$p\left(y_{1:M}^t \mid x_{1:M}^t, \mathcal{D}^c, \theta\right) \approx \mathcal{N}\left(y_{1:M}^t \mid \mu_y, \Sigma_y\right). \tag{3.12}$$

This is inspired by GPs which also define a Gaussian likelihood. Maximizing this expression yields the optimal solution $\mu_y = \tilde{\mu}_y, \Sigma_y = \tilde{\Sigma}_y$, with $\tilde{\mu}_y$ and $\tilde{\Sigma}_y$ being the first and second moments of the true posterior predictive distribution. This is a well-known result known as *moment matching*, a popular variant of deterministic approximate inference used, e.g., in Deisenroth et al. [Dei11b] and Becker et al. [Bec19]. $\tilde{\mu}_y$ and $\tilde{\Sigma}_y$ are functions of the moments μ_z and σ_z^2 of the latent posterior $p\left(z \mid \mathcal{D}^c, \theta\right)$ which motivates the following decoder parameterization:

$$\mu_y = \operatorname{dec}_{\mu_y,\theta}\left(\mu_z, \sigma_z^2, x_m^t\right), \quad \sigma_y^2 = \operatorname{dec}_{\sigma_y^2,\theta}\left(\mu_z, \sigma_z^2, x_m^t\right), \quad \Sigma_y = \operatorname{diag}\left(\sigma_y^2\right). \tag{3.13}$$

Here, μ_z and σ_z^2 are given by the BA Eqs. (3.10). Note that we define the Gaussian approximation to be factorized w.r.t. individual y_m^t, an assumption which simplifies the architecture but could be dropped if a more expressive model was required. This decoder can be interpreted as a "moment matching network", computing the moments of y given the moments of z. Indeed, in contrast to decoder networks of CLV-based NP architectures as defined in Eq. (3.5), it operates on the moments μ_z and σ_z^2 of the latent distribution instead of on samples z which allows to evaluate this approximation in a deterministic manner. In this sense, the resulting model is akin to the CNP which defines a deterministic, conditional model with a decoder operating on the mean-aggregated latent observation $\bar{r}$. However, BA-based models trained in this deterministic manner still benefit from BA's ability to accurately quantify latent parameter uncertainty which yields significantly improved predictive likelihoods. In what follows, we refer to this approximation scheme as direct parameter-based (PB) likelihood optimization.

3.4.3.3 Discussion

The concrete choice of likelihood approximation or, equivalently, model architecture depends mainly on the intended use-case. Sampling-based models are generally more expressive as they can represent complex, i.e., structured, non-Gaussian, posterior predictive distributions. Moreover, they yield true function samples while deterministic models only allow approximate function samples through auto-regressive (AR) sampling schemes. Nevertheless, deterministic models exhibit several computational advantages. They yield direct probabilistic predictions in a single forward pass, while the predictions of sampling-based methods are only defined through averages over multiple function samples and hence require multiple forward passes. Likewise, evaluating the MC-based likelihood approximation Eq. (3.3) during training requires to draw multiple (K) latent samples z. While the VI likelihood approximation Eq. (3.2) can be optimized on a single function sample per training step through stochastic gradient descent [Bis06], it has the disadvantage that it requires to feed target sets $\mathcal{D}^t$ through the encoder which can impede the training for small context sets $\mathcal{D}^c$ as discussed in detail in App. A.2.

3.5 Empirical Evaluation

We present experiments to compare the performances of BA and of MA in NP-based models. To provide a complete picture, we evaluate all combinations of likelihood approximations (PB/deterministic Eq. (3.12), VI Eq. (3.2), MC Eq. (3.3)) and aggregation methods (BA Eq. (3.10), MA Eq. (3.7)), resulting in six different model architectures, cf. Fig. A.1 in App. A.5.2. Two of these architectures correspond to existing members of the NP family: MA + deterministic is equivalent to the CNP [Gar18b], and MA + VI corresponds to the Latent-Path NP (LP-NP) [Gar18c], i.e., the NP without a deterministic path. We further evaluate the Attentive Neural Process (ANP) [Kim19], which employs a hybrid approach, combining LP-NP with a cross-attention mechanism in a parallel deterministic path[1], as well as an NP-architecture using MA with a self-attentive (SA) encoder network. Note that BA can also be used in hybrid models like ANP or in combination with SA, an idea we leave for future research. In App. A.4 we discuss NP-based regression in relation to other methods for (scalable) probabilistic regression.

The performance of NP-based models depends heavily on the encoder and decoder network architectures as well as on the latent space dimensionality d_z. To assess the influence of the aggregation mechanism independently from all other confounding factors, we consistently optimize the encoder and decoder network architectures, the latent-space dimensionality d_z, as well as the learning rate of the Adam optimizer [Kin15], *independently for all model architectures and for all experiments* using the Optuna [Aki19] framework, cf. App. A.5.3. If not stated differently, we report performance in terms of the mean posterior predictive log-likelihood over 256 test tasks with 256 data points each, conditioned on context sets containing $N \in \{0, 1, \dots, N_{\max}\}$ data points (cf. App. A.5.4). For sampling-based methods (VI, MC, ANP), we report the joint log-likelihood over the test sets using a Monte-Carlo approximation with 25 latent samples, cf. App. A.5.4. We average the resulting log-likelihood values over 10 training runs with different random seeds

[1] For ANP, we use original code from https://github.com/deepmind/neural-processes

and report 95% confidence intervals. We publish source code to reproduce the experimental results online.[1]

3.5.1 GP Samples

We evaluate the architectures on synthetic functions drawn from GP priors with different kernels (RBF, weakly periodic, Matern-5/2), as proposed by Gordon et al. [Gor20], cf. App. A.5.1. We generate a new batch of functions for each training epoch. The results (Fig. 3.4 and Tab. 3.1) show that BA consistently outperforms MA, independent of the model architecture.

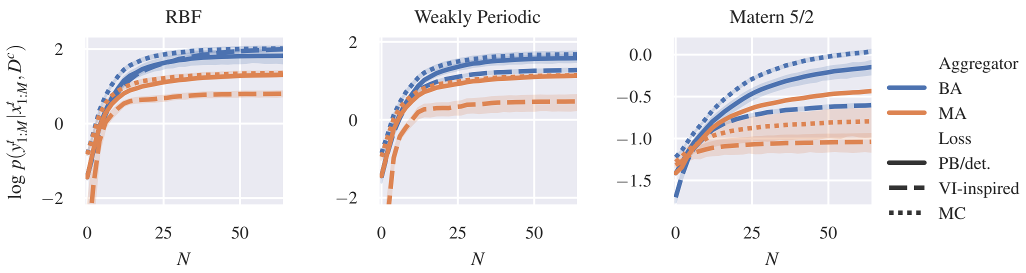


Figure 3.4: Posterior predictive log-likelihood on functions drawn from GP priors with RBF, weakly periodic, and Matern-5/2 kernels in dependence of the context size N. BA consistently outperforms MA, independent of the likelihood approximation, with MC being the most expressive choice. PB represents an efficient, deterministic alternative, while the VI approximation tends to perform worst, in particular for small N.

Table 3.1: Posterior predictive log-likelihood on functions drawn from GP priors with RBF, weakly periodic, and Matern-5/2 kernels, averaged over context sets with $N \in \{0, 1, \dots, 64\}$ points. Cf. Fig. 3.4 for a discussion.

		BA	MA
RBF GP	PB/det.	$\mathbf{1.37 \pm 0.15}$	0.94 ± 0.04
	VI	$\mathbf{1.40 \pm 0.04}$	0.45 ± 0.12
	MC	$\mathbf{1.62 \pm 0.05}$	1.07 ± 0.05
	ANP		0.98 ± 0.02
Weakly Periodic GP	PB/det.	$\mathbf{1.13 \pm 0.08}$	0.76 ± 0.02
	VI	$\mathbf{0.89 \pm 0.03}$	0.07 ± 0.14
	MC	$\mathbf{1.30 \pm 0.06}$	0.85 ± 0.04
	ANP		1.02 ± 0.02
Matern-5/2 GP	PB/det.	$\mathbf{-0.50 \pm 0.07}$	-0.68 ± 0.01
	VI	$\mathbf{-0.79 \pm 0.01}$	-1.09 ± 0.10
	MC	$\mathbf{-0.33 \pm 0.01}$	-0.90 ± 0.15
	ANP		$\mathbf{0.25 \pm 0.02}$

[1] https://github.com/boschresearch/bayesian-context-aggregation

Interestingly, despite employing a factorized Gaussian approximation, our deterministic PB approximation performs at least on-par with the traditional VI approximation which tends to perform particularly poorly for small context sets, reflecting the intricacies discussed in Sec. 3.4.3. As expected, the MC approximation yields the best results in terms of predictive performance, as it is more expressive than the deterministic approaches and does not share the problems of the VI approach. As shown in Tab. 3.2 and Tab. A.3, App. A.6, our proposed PB likelihood approximation is much cheaper to evaluate compared to both sampling-based approaches which require multiple forward passes per prediction.

Table 3.2: Relative evaluation runtimes and number of parameters of the optimized network architectures on RBF GP. Also cf. Tab. A.3.

		BA	MA
Runtime	PB/det.	1	1.4
	VI	18	25
	MC	32	27
#Parameters	PB/det.	72k	96k
	VI	63k	77k
	MC	122k	153k

We further observe that BA tends to require smaller encoder and decoder networks as it is more efficient at propagating context information to the latent state as discussed in Sec. 3.4.1. The hybrid ANP approach is competitive only on the Matern-5/2 function class. Yet, we refer the reader to Tab. A.4, App. A.6, demonstrating that the attention mechanism greatly improves performance in terms of MSE.

3.5.2 Quadratic Functions

We further seek to study the performance of BA with very limited amounts of training data. To this end, we consider two quadratic function classes, each parameterized by three real parameters from which we generate limited numbers L of training tasks. The first function class is defined on a one-dimensional domain, i.e., $x \in \mathbb{R}$, and we choose $L = 64$, while the second function class, as proposed by Perrone et al. [Per18], is defined on $x \in \mathbb{R}^3$ with $L = 128$, cf. App. A.5.1. As shown in Tab. 3.3, BA again consistently outperforms MA, often by considerably large margins, underlining the efficiency of our Bayesian approach to aggregation in the regime of little training data.

Table 3.3: Posterior predictive log-likelihood on 1D and 3D quadratic functions with limited numbers L of training tasks, averaged over context sets with $N \in \{0, 1, \ldots, 20\}$ data points. BA outperforms MA by considerable margins in this regime of little training data.

		BA	MA
Quadratic 1D, $L = 64$	PB/det.	$\mathbf{1.42 \pm 0.20}$	0.47 ± 0.25
	VI	$\mathbf{1.48 \pm 0.05}$	-0.32 ± 0.55
	MC	$\mathbf{1.71 \pm 0.23}$	1.27 ± 0.06
	ANP		0.69 ± 0.08
Quadratic 3D, $L = 128$	PB/det.	$\mathbf{-2.46 \pm 0.12}$	-2.73 ± 0.10
	VI	$\mathbf{-2.53 \pm 0.07}$	-3.45 ± 0.12
	MC	$\mathbf{-1.79 \pm 0.07}$	-2.14 ± 0.05
	ANP		-3.08 ± 0.02

On the 1D task, all likelihood approximations perform approximately on-par in combination with BA, while MC outperforms both on the more complex 3D task. Fig. 3.5 compares prediction qualities.

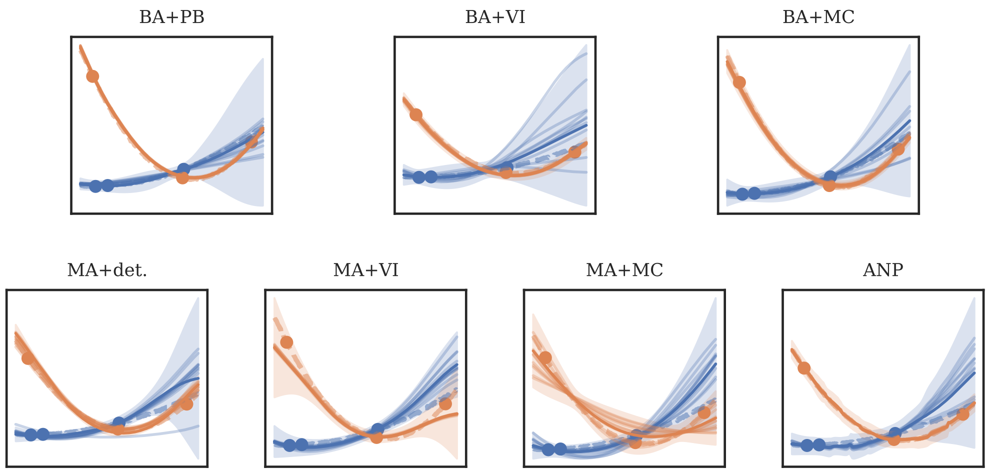


Figure 3.5: Predictions on two instances (dashed lines) of the 1D quadratic function class, given $N = 3$ context data points (circles). We show mean and standard deviation predictions (solid line, shaded area), and 10 function samples (AR samples for deterministic methods). Cf. also App. A.6.

3.5.3 Dynamics of a Furuta Pendulum

We study BA on a realistic dataset given by the simulated dynamics of a rotary inverted pendulum, better known as the Furuta pendulum [Fur92], which is a highly non-linear dynamical system, consisting of an actuated arm rotating in the horizontal plane with an attached pendulum rotating freely in the vertical plane (Fig. 3.6), parameterized by two masses, three lengths, and two damping constants.

Figure 3.6: A Furuta pendulum as distributed by the Quanser company (QUBE™ Servo 2). Image source: https://www.quanser.com

The regression task is defined as the one-step-ahead prediction of the four-dimensional system state with a step-size of $\Delta t = 0.1$ s, as detailed in App. A.5.1. The results (Tab. 3.4) show that BA improves predictive performance also on complex, non-synthetic regression tasks with higher-dimensional input- and output spaces.

Table 3.4: Posterior predictive log-likelihood on the dynamics of a Furuta pendulum, averaged over context sets with $N \in \{0, 1, \dots, 20\}$ state transitions. BA performs favorably on this real-world task.

		BA	MA
Furuta Dynamics	PB/det.	$\mathbf{7.50 \pm 0.27}$	7.06 ± 0.12
	VI	$\mathbf{7.32 \pm 0.18}$	5.57 ± 0.21
	MC	$\mathbf{8.25 \pm 0.33}$	7.55 ± 0.24
	ANP		4.74 ± 0.16

Further, they are consistent with our previous findings regarding the likelihood approximations, with MC being strongest in terms of predictive likelihood, followed by our efficient deterministic alternative PB.

3.5.4 2D Image Completion

We consider a 2D image completion experiment where the inputs x are pixel locations in images showing handwritten digits, and we regress onto the corresponding pixel intensities y, cf. App. A.6. Interestingly, we found that architectures without deterministic paths were not able to solve this task reliably which is why we only report results for deterministic models. As shown in Tab. 3.5, BA improves performance in comparison to MA by a large margin.

Table 3.5: Predictive log-likelihood on a 2D image completion task on MNIST, averaged over $N \in \{0, 1, \dots, 392\}$ context pixels.

		BA	MA
2D Image Completion	PB/det.	$\mathbf{2.75 \pm 0.20}$	2.05 ± 0.36
	ANP		1.62 ± 0.03

This highlights that BA's ability to quantify the information content of a context tuple is particularly beneficial on this task, as, e.g., pixels in the middle area of the images typically convey more information about the identity of the digit than pixels located near the borders.

3.5.5 Self-attentive Encoders

Another interesting baseline for BA is MA, combined with a self-attention (SA) mechanism in the encoder. Indeed, similar to BA, SA yields non-uniform weights for the latent observations r_n, where a given weight is computed from some form of pairwise spatial relationship with all other latent observations in the context set (cf. App. A.3 for a detailed discussion). As BA's weight for r_n only depends on (x_n, y_n) itself, BA is computationally more efficient: SA scales like $\mathcal{O}(N^2)$ in the number N of context tuples while BA scales like $\mathcal{O}(N)$, and, furthermore, SA does not allow for efficient incremental updates while this is possible for BA, cf. Eq. (3.11). Tabs. 3.6 and 3.7 show a comparison of BA with MA in combination with various different SA mechanisms in the encoder.

We emphasize that we compare against BA in its vanilla form, i.e., BA does not use SA in the encoder. The results show that Laplace SA and dot-product SA do not improve predictive performance compared to vanilla MA, while multihead SA yields significantly better results. Nevertheless, vanilla BA still performs better or at least on-par and is computationally more efficient. While being out of the scope of this work, according to these results, a combination of BA with SA seems promising if computational disadvantages can be accepted in favour of increased predictive performance, cf. App. A.3.

3.6 Conclusion and Outlook

We proposed a novel Bayesian Aggregation (BA) method for NP-based models, combining context aggregation and hidden parameter inference in one holistic mechanism which enables efficient handling of task ambiguity. BA is conceptually simple, compatible with existing NP-based model architectures, and consistently improves performance compared to traditional mean aggregation. It introduces only marginal computational overhead, simplifies the architectures in comparison to existing CLV models (no $\bar{r}$-to-z-networks), and tends to require less complex encoder and decoder network architectures. Our experiments further demonstrate that the VI likelihood approximation traditionally used to train NP-based models should be abandoned in favor of a MC-based approach, and that our proposed PB likelihood approximation represents an efficient deterministic alternative with strong predictive performance. We believe that a range of existing models, e.g., the ANP or NPs with self-attentive encoders, can benefit from BA, especially when a reliable quantification of uncertainty is crucial. Also, more complex Bayesian aggregation models are conceivable, opening interesting avenues for future research.

Table 3.6: Comparison of the posterior predictive log-likelihood of our BA with traditional MA, combined with a self-attention (SA) mechanism in the encoder (BA does not use an SA mechanism), using the PB likelihood approximation. We provide results for Laplace SA (L-SA), dot-product SA (DP-SA), and mulihead SA (MH-SA) and repeat the results for BA and MA without SA ("no SA"). While L-SA and DP-SA do not increase predictive performance compared to MA without SA, MH-SA results in significant improvements. Nevertheless, vanilla BA still performs better or at least on-par, while being computationally more efficient.

		BA + PB	MA + PB
RBF GP	no SA	$\mathbf{1.37 \pm 0.15}$	0.94 ± 0.04
	L-SA		0.74 ± 0.06
	DP-SA		0.89 ± 0.04
	MH-SA		$\mathbf{1.46 \pm 0.14}$
Weakly Periodic GP	no SA	$\mathbf{1.13 \pm 0.08}$	0.76 ± 0.02
	L-SA		0.59 ± 0.02
	DP-SA		0.71 ± 0.02
	MH-SA		$\mathbf{1.13 \pm 0.15}$
Matern-5/2 GP	no SA	$\mathbf{-0.50 \pm 0.07}$	-0.68 ± 0.01
	L-SA		-1.03 ± 0.01
	DP-SA		-0.76 ± 0.01
	MH-SA		-0.64 ± 0.01
Quadratic 1D, $L = 64$	no SA	$\mathbf{1.42 \pm 0.20}$	0.47 ± 0.25
	L-SA		0.15 ± 0.32
	DP-SA		0.47 ± 0.24
	MH-SA		$\mathbf{1.49 \pm 0.11}$
Quadratic 3D, $L = 128$	no SA	$\mathbf{-2.46 \pm 0.12}$	-2.73 ± 0.10
	L-SA		-2.94 ± 0.41
	DP-SA		-2.95 ± 0.13
	MH-SA		$\mathbf{-2.13 \pm 0.25}$
Furuta Dynamics	no SA	$\mathbf{7.50 \pm 0.27}$	7.06 ± 0.12
	L-SA		7.13 ± 0.12
	DP-SA		7.04 ± 0.20
	MH-SA		$\mathbf{7.40 \pm 0.46}$

Table 3.7: Comparison of the posterior predictive log-likelihood of our BA with traditional MA, combined with a self-attention (SA) mechanism in the encoder (BA does not use an SA mechanism), using the MC likelihood approximation. We provide results for Laplace SA (L-SA), dot-product SA (DP-SA), and mulihead SA (MH-SA) and repeat the results for BA and MA without SA ("no SA"). While L-SA and DP-SA do not increase predictive performance compared to MA without SA, MH-SA results in significant improvements. Nevertheless, vanilla BA still performs better or at least on-par, while being computationally more efficient.

		BA + MC	MA + MC
RBF GP	no SA	$\mathbf{1.62 \pm 0.05}$	1.07 ± 0.05
	L-SA		0.93 ± 0.05
	DP-SA		0.98 ± 0.03
	MH-SA		1.44 ± 0.09
Weakly Periodic GP	no SA	$\mathbf{1.30 \pm 0.06}$	0.85 ± 0.04
	L-SA		0.77 ± 0.03
	DP-SA		0.82 ± 0.03
	MH-SA		$\mathbf{1.29 \pm 0.04}$
Matern-5/2 GP	no SA	$\mathbf{-0.33 \pm 0.01}$	-0.90 ± 0.15
	L-SA		-0.80 ± 0.02
	DP-SA		-0.86 ± 0.01
	MH-SA		-0.59 ± 0.03
Quadratic 1D, $L = 64$	no SA	$\mathbf{1.71 \pm 0.23}$	1.27 ± 0.06
	L-SA		1.19 ± 0.09
	DP-SA		1.32 ± 0.14
	MH-SA		$\mathbf{1.66 \pm 0.12}$
Quadratic 3D, $L = 128$	no SA	$\mathbf{-1.79 \pm 0.07}$	-2.14 ± 0.05
	L-SA		-2.19 ± 0.11
	DP-SA		-2.18 ± 0.07
	MH-SA		$\mathbf{-1.71 \pm 0.05}$
Furuta Dynamics	no SA	$\mathbf{8.25 \pm 0.33}$	7.55 ± 0.24
	L-SA		7.80 ± 0.13
	DP-SA		7.67 ± 0.14
	MH-SA		$\mathbf{8.39 \pm 0.20}$

4 Accurate Task Posterior Inference with Gaussian Mixture Models

In Sec. 3, we introduced Bayesian context aggregation (BA), a novel context aggregation scheme for NP-based BML models. While BA improves the parameterization of the task posterior approximation, it still operates within the standard NP approximate inference scheme with amortized Gaussian mean-field approximations and reparameterized Euclidean gradients (Sec. 2.2.4) [Gar18c].

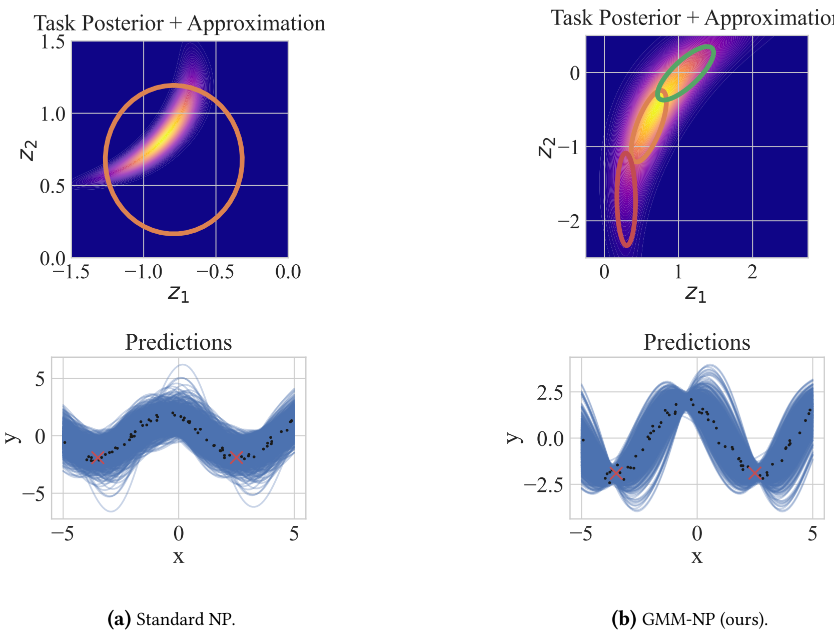


Figure 4.1: Comparison of standard NPs [Gar18c] and our GMM-NP architecture. We train instances of these architectures with a $d_z = 2$ dimensional hypothesis space $\mathcal{Z}$ on a meta dataset $\mathcal{D}^{\text{meta}}$ generated from sinusoidal functions whose amplitudes and phases vary from task to task. **Top row:** True (unnormalized) task posterior distributions $p(\boldsymbol{z} \mid \mathcal{D}^{c,*}, \theta)$ (contours), given a context set $\mathcal{D}^{c,*} \subset \mathcal{D}^*$ with two examples from an unseen test dataset $\mathcal{D}^*$, along with the corresponding task posterior approximations (Gaussian ellipses). **Bottom row:** Samples from the corresponding predictive distributions (blue), test data $\mathcal{D}^*$ (black dots), and context data $\mathcal{D}^{c,*}$ (red crosses). **Panel (a)**: Standard NPs use Gaussian mean-field VI, which yields a crude task posterior approximation and results in suboptimal predictions. **Panel (b)**: GMM-NP uses an expressive full-covariance Gaussian mixture task posterior approximation (we use three mixture components in this example), resulting in significantly improved predictive performance.

In this chapter, we investigate the impact of these design choices and show that they lead to a significant variational inference gap and, consequently, to suboptimal solutions for the MMLE $\hat{\theta}$ (Sec. 2.1.6.4). This observation leads us to develop GMM-NP, a novel NP variant that uses a non-amortized mixture of full-covariance Gaussians to approximate the task posterior (Fig. 4.1). We demonstrate that this approach yields significantly improved predictive performance, and that it can be implemented efficiently following recent work combining natural gradients and information-geometric trust regions (Sec. 2.1.6.3) [Are23, Lin20].

The remainder of this chapter has been published as [Vol23] Volpp, Michael; Dahlinger, Philipp; Becker, Philipp; Daniel, Christian, and Neumann, Gerhard: "Accurate Bayesian Meta-Learning by Accurate Task Posterior Inference". In: International Conference on Learning Representations *(2023). Reprinted with permission from the authors.*

4.1 Introduction

Driven by algorithmic advances in the field of deep learning (DL) and the availability of increasingly powerful GPU-assisted hardware, the field of machine learning achieved a plethora of impressive results in recent years [Par18, Rad19, Mni15b]. These were enabled to a large extent by the availability of huge datasets, which enables training expressive deep neural network (DNN) models. In practice, e.g., in industrial settings, such datasets are unfortunately rarely available, rendering standard DL approaches futile. Nevertheless, it is often the case that similar tasks arise repeatedly, such that the number of context examples on a novel target task is typically relatively small, but the joint *meta-dataset* of examples from all tasks accumulated over time can be massive, s.t. powerful inductive biases can be extracted using *meta-learning* [Hos22]. While these inductive biases allow restricting predictions to only those compatible with the meta-data, there typically remains *epistemic uncertainty* due to task ambiguity, as the context data is often not informative enough to identify the target task exactly. *Bayesian meta-learning* (BML) aims at an accurate quantification of this uncertainty, which is crucial for applications like active learning, Bayesian optimization [Sha16], model-based reinforcement learning [Chu18], robotics [Dei11a], and in safety-critical scenarios.

A prominent BML approach is the Neural Process (NP) [Gar18c] which employs a DNN-based conditional latent variable (CLV) model, in which the Bayesian belief about the target task is encoded in a factorized Gaussian task posterior (TP) approximation, and inference is amortized over tasks using set encoders [Zah17]. This architecture can be optimized efficiently using variational inference (VI) with standard, reparameterized gradients [Kin13]. A range of modifications, such as adding deterministic, attentive, computation paths [Kim19], or Bayesian set encoders [Vol21], have been proposed in recent years to improve predictive performance. Interestingly, the VI scheme with an amortized, factorized Gaussian TP, optimized using standard gradients, remains largely unaltered. Yet, it is well known that (i) the factorized Gaussian assumption rarely holds in

Bayesian learning [Mac03, Wil20], (ii) amortized inference can yield suboptimal posterior approximations [Cre18], and (iii) natural gradients are superior to standard gradients for VI in terms of optimization efficiency and robustness [Kha18a].

Building on these insights and on recent advances in VI [Lin20, Are23], we propose GMM-NP, a novel NP-based BML algorithm that employs (i) a full-covariance Gaussian mixture (GMM) TP approximation, optimized in a (ii) non-amortized fashion, using (iii) robust and efficient trust region natural gradient (TRNG)-VI. We demonstrate through extensive empirical evaluations and ablations that our approach yields tighter evidence lower bounds, more efficient model optimization, and, thus, markedly improved predictive performance, outperforming the state-of-the-art both in terms of epistemic uncertainty quantification and accuracy. Notably, GMM-NP does not require complex architectural modifications, which shows that accurate TP inference is crucial for accurate BML, an insight we believe will be valuable for future research.

4.2 Related Work

Multitask learning aims to leverage inductive biases learned on a meta-dataset of similar tasks for improved data efficiency on unseen target tasks of similar structure. Notable variants include transfer-learning [Zhu21], that refines and combines pre-trained models [Gol17, Kri12], and meta-learning [Sch87, Thr98, Vil05, Hos22], which makes the multi-task setting explicit in the model design by formulating fast adaptation mechanisms in order to learn *how* to solve tasks with little context data (“few-shot learning”). A plethora of architectures were studied in the literature, including learner networks that adapt model parameters [Ben91, Sch92, Rav17], memory-augmented DNNs [San16], early instances of Bayesian meta-models [Edw17, Hew18], and algorithms that that make use of learned measures of task similarity [Koc15, Vin16, Sne17].

Arguably the most prominent meta-learning approaches are the Model-agnostic Meta-learning (MAML) and the Neural Process (NP) model families, due to their generality and flexibility. While the original MAML [Fin17] and Conditional NP [Gar18b] formulations do not explicitly model the epistemic uncertainty arising naturally in few-shot settings due to task ambiguity, both model families were extended to fully Bayesian meta-learning (BML) algorithms that explicitly infer the TP based on a CLV formulation [Hes00, Bak03]. Important representatives are Probabilistic MAML [Gra18, Fin18] and Bayesian MAML [Kim18], as well as several NP-based BML approaches that inspire our work. These include the Standard NP [Gar18c], which was extended by attentive computation paths to avoid underfitting [Kim19], or by Bayesian set encoders [Zah17, Wag19, Vol21] for improved handling of task ambiguity, as well as by hierarchical [Wan20a], bootstrapped [Lee20], or graph-based [Lou19] latent distributions. While the original NP formulation employs an amortized, reparameterized, stochastic gradient VI objective [Kin13, Rez14], Monte-Carlo (MC)-based objective functions were also studied [Gor19, Vol21].

From a more general perspective, VI emerged as a central tool in many areas of probabilistic machine learning, which require tractable approximations of intractable probability distributions, typically arising as the posterior in Bayesian models [Gel13, Kol09, Nea96, Wil20]. While early approaches [Att00] allow analytic updates, more complex algorithms employ stochastic gradients w.r.t. the variational parameters [Ran14, Kin13, Blu15]. Such approaches are straightforward to implement and computationally efficient for factorized Gaussian variational distributions, but ignore the information geometry of the loss landscape, leading to suboptimal convergence rates [Kha18a]. Natural gradient (NG)-VI [Ama98] alleviates this problem and recent work [Hof13, Win05, Kha18a, Kha18b] successfully applies this idea at scale to complex models, requiring only first-order gradient information [Lin19]. Further extensions enable NG-VI for structured variational distributions such as mixture models by decomposing the NG update into individual updates per mixture component [Are18, Lin20] which, in combination with trust region (TR) step size control [Abd15, Are23], yields robust and efficient VI algorithms for versatile and highly expressive variational distributions such as Gaussian mixture models (GMMs).

4.3 Preliminaries

We now briefly recap the TRNG-VI algorithm [Lin20, Are23] as well as the NP model [Gar18c], which form the central building blocks of our GMM-NP model.

4.3.1 Trust Region Natural Gradient VI with Gaussian Mixture Models

4.3.1.1 Variational Inference

We consider a probability distribution $p\left(\boldsymbol{z}\right)$ over a random variable $\boldsymbol{z} \in \mathbb{R}^{d_z}$, which is intractable in the sense that we know it only up to some normalization constant Z, i.e., $p\left(\boldsymbol{z}\right) = \tilde{p}\left(\boldsymbol{z}\right)/Z$ with $Z = \int p\left(\boldsymbol{z}\right) \mathrm{d}z$ and tractable $\tilde{p}(\boldsymbol{z})$. We seek to approximate $p\left(\boldsymbol{z}\right)$ by a tractable distribution $q_{\boldsymbol{\phi}}\left(\boldsymbol{z}\right)$, parameterized by $\boldsymbol{\phi}$. Variational inference (VI) frames this task as the minimization w.r.t. $\boldsymbol{\phi}$ of the reverse Kullback-Leiber (KL) divergence [Kul51]

$$\mathrm{KL}\left[q_{\boldsymbol{\phi}} \| p\right] \equiv -\mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z})}\left[\log \frac{\tilde{p}\left(\boldsymbol{z}\right)}{q_{\boldsymbol{\phi}}\left(\boldsymbol{z}\right)}\right] + \log Z \equiv -\mathcal{L}\left(\boldsymbol{\phi}\right) + \log Z, \tag{4.1}$$

where we introduced evidence lower bound (ELBO) $\mathcal{L}\left(\boldsymbol{\phi}\right)$. As Z is independent of $\boldsymbol{\phi}$, minimizing the KL divergence is equivalent to maximizing the ELBO.

4.3.1.2 Natural Gradients

A standard approach employs stochastic, reparameterized gradients [Kin13] w.r.t. $\boldsymbol{\phi}$ for optimization. While this is computationally efficient, it ignores the geometry of the statistical manifold defined by the set of probability distributions $q_{\boldsymbol{\phi}}$, which can lead to suboptimal convergence rates [Kha18a]. A more efficient solution is to perform updates in the *natural gradient* (NG) direction, i.e., the direction of steepest ascent w.r.t. the Fisher information metric [Ama98]. State-of-the-art approaches estimate the NG from first-order gradients of $p(\boldsymbol{z})$ by virtue of Stein's lemma [Lin19], yielding efficient NG-VI algorithms that scale to complex problems [Kha18b, Lin20, Are23].

4.3.1.3 Trust Regions

Selecting appropriate step sizes for updates in $\boldsymbol{\phi}$ can be intricate, which is why Abdolmaleki et al. [Abd15] propose a (zero-order) algorithm that incorporates a *trust region* constraint of the form $\mathrm{KL}\left[q_{\boldsymbol{\phi}} \| q_{\boldsymbol{\phi}_{\mathrm{old}}}\right] \leq \varepsilon$, which restricts the updates in distribution space and can be enforced with manageable computational overhead (a scalar, convex optimization problem in the Lagrangean parameter for the constraint). As shown by Arenz et al. [Are23], such trust regions can easily be combined with gradient information, and allow more aggressive updates in comparison to setting the step size directly, while still ensuring robust convergence.

4.3.1.4 VI with Gaussian Mixture Models

The quality of the approximation depends on the expressiveness of the distribution family $q_{\boldsymbol{\phi}}$. In settings where p corresponds to the Bayesian posterior of complex latent variable models [Mac03, Wil20], simple Gaussian approximations do not yield satisfactory results, as p typically is multimodal. In such cases, Gaussian mixture models (GMMs) are an appealing choice, as they provide cheap sampling, evaluation, and marginalization while allowing expressive approximations [Are18]. However, a naive application of VI is futile because gradients are coupled between GMM components, leading to computationally intractable updates. Fortunately, Arenz et al. [Are18] and Lin et al. [Lin20] show that updating the components and weights individually is possible, while preventing a collapse of the approximation onto a single posterior mode. This leads to two state-of-the-art algorithms for NG-VI with variational GMMs, that differ most notably in the way the step sizes for the updates are controlled: iBayes-GMM [Lin20], which directly sets step sizes for the updates, and TRNG-VI [Are23], which employs trust regions for more efficient and robust convergence.

4.3.2 Bayesian Meta-Learning with Neural Processes

4.3.2.1 The Multitask Latent Variable Model

We aim to fit a generative model to a meta-dataset $\mathcal{D} = \mathcal{D}_{1:L}$, consisting of regression *tasks* $\mathcal{D}_\ell = \left\{\boldsymbol{x}_{\ell,1:N}, \boldsymbol{y}_{\ell,1:N}\right\}$ with inputs $\boldsymbol{x}_{\ell,n} \in \mathbb{R}^{d_x}$ and corresponding evaluations $\boldsymbol{y}_{\ell,n} \in \mathbb{R}^{d_y}$ of unknown functions f_ℓ, i.e., $\boldsymbol{y}_{\ell,n} = f_\ell\left(\boldsymbol{x}_{\ell,n}\right) + \boldsymbol{\varepsilon}_n$, where $\boldsymbol{\varepsilon}_n$ denotes (possibly heteroskedastic) noise. Tasks are assumed to share statistical structure as formalized in the multi-task CLV model

(Fig. 4.2), defining the joint probability distribution

$$p_\theta\left(\boldsymbol{y}_{1:L,1:N}, \boldsymbol{z}_{1:L} \mid \boldsymbol{x}_{1:L,1:N}\right) = \prod_{\ell,n} p_\theta\left(\boldsymbol{y}_{\ell,n} \mid \boldsymbol{x}_{\ell,n}, \boldsymbol{z}_\ell\right) p\left(\boldsymbol{z}_\ell\right), \tag{4.2}$$

where $\boldsymbol{z}_\ell \in \mathbb{R}^{d_z}$ denote latent task descriptors and $\boldsymbol{\theta}$ denotes task-global parameters that capture shared statistical structure.

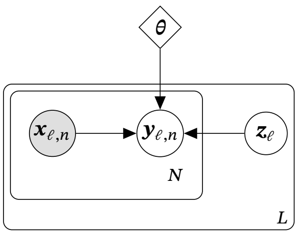


Figure 4.2: Multitask CLV model with task-local latent variables $\boldsymbol{z}_\ell$ and a global latent variable θ, capturing structure shared between tasks.

Having observed context data $\mathcal{D}_*^c = \{\boldsymbol{x}_{*,1:M_*}^c, \boldsymbol{y}_{*,1:M_*}^c\} \subset \mathcal{D}_*$ from a target task $\mathcal{D}_*$, predictions are provided in terms of the marginal predictive distribution

$$p_\theta\left(\boldsymbol{y}_{*,1:N_*} \mid \boldsymbol{x}_{*,1:N_*}, \mathcal{D}_*^c\right) = \int \prod_n p_\theta\left(\boldsymbol{y}_{*,n} \mid \boldsymbol{x}_{*,n}, \boldsymbol{z}_*\right) p_\theta\left(\boldsymbol{z}_* \mid \mathcal{D}_*^c\right) \mathrm{d}z_*, \tag{4.3}$$

with the *task posterior* (TP) distribution

$$p_\theta\left(\boldsymbol{z}_* \mid \mathcal{D}_*^c\right) = \frac{\prod_m p_\theta(\boldsymbol{y}_{*,m}^c \mid \boldsymbol{x}_{*,m}^c, \boldsymbol{z}_*) p\left(\boldsymbol{z}_*\right)}{p_\theta\left(\mathcal{D}_*^c\right)}. \tag{4.4}$$

4.3.2.2 The Neural Process

In its standard formulation, the Neural Process (NP) [Gar18c] defines a factorized Gaussian likelihood

$$p_\theta\left(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{z}\right) \equiv \mathcal{N}\left(\boldsymbol{y} \mid \mathrm{dec}_\theta^\mu\left(\boldsymbol{x}, \boldsymbol{z}\right), \mathrm{diag}\left(\sigma_\mathrm{n}^2\right)\right), \tag{4.5}$$

where the *decoder* dec_θ^μ is a DNN with weights $\boldsymbol{\theta}$, and observation noise variance σ_n^2. As the TP is intractable for this likelihood choice, NP computes a factorized Gaussian approximation

$$q_\phi\left(\boldsymbol{z}_* \mid \mathcal{D}_*^c\right) \equiv \mathcal{N}(\boldsymbol{z}_* \mid \mathrm{enc}_\phi^\mu\left(\mathcal{D}_*^c\right), \mathrm{diag}(\mathrm{enc}_\phi^\sigma\left(\mathcal{D}_*^c\right))) \tag{4.6}$$

with *deep set encoders* [Zah17, Wag19] enc_ϕ^μ, enc_ϕ^σ, parameterized by $\boldsymbol{\phi}$. The parameters $\boldsymbol{\Phi} \equiv (\boldsymbol{\theta}, \boldsymbol{\phi})$ are optimized jointly on the meta-data by stochastic gradient ascent on the ELBO $\sum_{\ell=1}^L \mathcal{L}_\ell\left(\boldsymbol{\Phi}\right)$

w.r.t. the approximate log marginal predictive likelihood defined by

$$\log q_{\boldsymbol{\Phi}}\left(\boldsymbol{y}_{\ell,1:N} \mid \boldsymbol{x}_{\ell,1:N}, \mathcal{D}_\ell^c\right) \equiv \log \int \prod_n p_\theta\left(\boldsymbol{y}_{\ell,n} \mid \boldsymbol{x}_{\ell,n}, \boldsymbol{z}_\ell\right) q_{\boldsymbol{\phi}}\left(\boldsymbol{z}_\ell \mid \mathcal{D}_\ell^c\right) \mathrm{d}z_\ell \tag{4.7}$$

$$\geq \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}_\ell|\mathcal{D}_\ell)}\left[\sum_n \log p_\theta\left(\boldsymbol{y}_{\ell,n} \mid \boldsymbol{x}_{\ell,n}, \boldsymbol{z}_\ell\right) + \log \frac{q_{\boldsymbol{\phi}}\left(\boldsymbol{z}_\ell \mid \mathcal{D}_\ell^c\right)}{q_{\boldsymbol{\phi}}\left(\boldsymbol{z}_\ell \mid \mathcal{D}_\ell\right)}\right] \equiv \mathcal{L}_\ell\left(\boldsymbol{\Phi}\right), \tag{4.8}$$

where $\mathcal{D}_\ell^c \subset \mathcal{D}_\ell$, and stochastic gradients w.r.t. $\boldsymbol{\phi}$ are estimated using the reparameterization trick [Kin13]. Note that NP *amortizes* inference (the variational parameters $\boldsymbol{\phi}$ are shared across tasks) and that it re-uses $q_{\boldsymbol{\phi}}\left(\boldsymbol{z}_\ell \mid \cdot\right)$ to compute the variational distribution $q_{\boldsymbol{\phi}}\left(\boldsymbol{z}_\ell \mid \mathcal{D}_\ell\right)$, taking advantage of its deep set encoder, which allows to condition it on datasets of arbitrary size.

4.4 Bayesian Meta-Learning with GMM Task Posteriors

4.4.1 Motivation

Our work is motivated by the observation that the current state-of-the-art approach for training NP-based BML models is suboptimal. Concretely, we identify three interrelated issues with the optimization objective Eq. (4.8):

(I1) **Expressivity of the Variational Distribution.** $q_{\boldsymbol{\phi}}$ is a (i) factorized, (ii) unimodal Gaussian distribution, (iii) amortized over tasks. In effect, this parameterization only allows crude approximations of the TP distribution [Mac03, Cre18].

(I2) **Optimization of the Variational Parameters.** (i) Naive gradients of Eq. (4.8), ignoring the information geometry of $q_{\boldsymbol{\phi}}$, with (ii) direct step size control are employed for optimization, yielding brittle convergence at suboptimal rates [Kha18a, Are23].

(I3) **Optimization of the Model Parameters.** Due to the suboptimal VI scheme (I1, I2), the TP approximation is poor, resulting in a loose ELBO Eq. (4.8). In effect, optimization w.r.t. the model parameters $\boldsymbol{\theta}$ is inefficient, cf. App. B.1.4.1 for a detailed discussion.

Armed with these insights, we develop a novel BML algorithm that is close in spirit to the NP but solves (I1-I3) through TRNG-VI with GMM TP approximations.

4.4.2 Model

Our algorithm builds on the standard multi-task CLV architecture Eq. (4.2) and retains the likelihood parameterization using a decoder DNN, $\mathrm{dec}_{\boldsymbol{\theta}}^{\mu}\left(\boldsymbol{x}, \boldsymbol{z}\right)$, as this allows for expressive BML models. Under this parameterization, the log marginal likelihood for a single task reads

$$\log p_\theta\left(\boldsymbol{y}_{\ell,1:N} \mid \boldsymbol{x}_{\ell,1:N}\right) = \log \int \prod_n p_\theta\left(\boldsymbol{y}_{\ell,n} \mid \boldsymbol{x}_{\ell,n}, \boldsymbol{z}_\ell\right) p\left(\boldsymbol{z}_\ell\right) \mathrm{d}z_\ell \equiv \log Z_\ell\left(\boldsymbol{\theta}\right), \tag{4.9}$$

where $Z_\ell(\boldsymbol{\theta})$ is the normalization constant of the TP

$$p_{\boldsymbol{\theta}}(\boldsymbol{z}_\ell \mid \mathcal{D}_\ell) = \frac{\tilde{p}_\ell(\boldsymbol{z}_\ell)}{Z_\ell(\boldsymbol{\theta})} \tag{4.10}$$

with

$$\tilde{p}(\boldsymbol{z}_\ell) \equiv \prod_n p_{\boldsymbol{\theta}}\left(\boldsymbol{y}_{\ell,n} \mid \boldsymbol{x}_{\ell,n}, \boldsymbol{z}_\ell\right) p(\boldsymbol{z}_\ell). \tag{4.11}$$

In contrast to Eq. (4.7), we do not condition the left hand side on a context set $\mathcal{D}_\ell^c$, which yields a tractable integrand $\tilde{p}(\boldsymbol{z}_\ell)$ that does not require further approximation. To tackle (I1), we approximate $p_{\boldsymbol{\theta}}(\boldsymbol{z}_\ell \mid \mathcal{D}_\ell)$ by an expressive variational GMM of the form

$$q_{\boldsymbol{\phi}_\ell}(\boldsymbol{z}_\ell) \equiv \sum_k w_{\ell,k} q_{\boldsymbol{\phi}_\ell}(\boldsymbol{z}_\ell \mid k) \equiv \sum_k w_{\ell,k} \mathcal{N}\left(\boldsymbol{z}_\ell \mid \boldsymbol{\mu}_{\ell,k}, \boldsymbol{\Sigma}_{\ell,k}\right), \tag{4.12}$$

with $\sum_k w_{\ell,k} = 1$. Note that we train individual GMMs with parameters $\boldsymbol{\phi}_\ell \equiv \left\{w_{\ell,k}, \boldsymbol{\mu}_{\ell,k}, \boldsymbol{\Sigma}_{\ell,k}\right\}$, $k \in \{1, \dots, K\}$ for each task ℓ, to not impair approximation quality by introducing inaccuracies through amortization.

4.4.3 Update Equations for the Variational Parameters

To ensure efficient and robust optimization of $\boldsymbol{\phi}_\ell$ (I2), we employ TRNG-VI as proposed by Arenz et al. [Are23], with the update equations

$$\boldsymbol{\Sigma}_{\ell,k,\text{new}} = \left[\frac{\eta}{\eta+1}\boldsymbol{\Sigma}_{\ell,k,\text{old}}^{-1} - \frac{1}{\eta+1}\boldsymbol{R}_{\ell,k}\right]^{-1}, \tag{4.13a}$$

$$\boldsymbol{\mu}_{\ell,k,\text{new}} = \boldsymbol{\Sigma}_{\ell,k,\text{new}}\left[\frac{\eta}{\eta+1}\boldsymbol{\Sigma}_{\ell,k,\text{old}}^{-1}\boldsymbol{\mu}_{\ell,k,\text{old}} + \frac{1}{\eta+1}\left(\boldsymbol{r}_{\ell,k} - \boldsymbol{R}_{\ell,k}\boldsymbol{\mu}_{\ell,k,\text{old}}\right)\right], \tag{4.13b}$$

$$w_{\ell,k,\text{new}} \propto \exp \rho_{\ell,k}, \tag{4.13c}$$

where $\boldsymbol{R}_{\ell,k}$, $\boldsymbol{r}_{\ell,k}$, and $\rho_{\ell,k}$ are defined as expectations that can be approximated from per-component samples using MC and require at most first-order gradients of $\tilde{p}(\boldsymbol{z}_\ell)$, which are readily available using standard automatic differentiation software [Mar15, Pas19]. Due to space constraints, we move details to App. B.1.1. The optimal value for the trust region parameter $\eta \geq 0$ is defined by a scalar convex optimization problem that can be solved efficiently by a bracketing search, which also ensures positive definiteness of the new covariance matrix $\boldsymbol{\Sigma}_{\ell,k,\text{new}}$.

4.4.4 Updates for the Model Parameters

To optimize the model parameters $\boldsymbol{\theta}$, we decompose the log marginal likelihood $\log Z_\ell(\boldsymbol{\theta})$ according to Eq. (4.1) as

$$\log Z_\ell(\boldsymbol{\theta}) = \mathbb{E}_{q_{\boldsymbol{\phi}_\ell}(\boldsymbol{z}_\ell)}\left[\sum_n \log p_{\boldsymbol{\theta}}\left(\boldsymbol{y}_{\ell,n} \mid \boldsymbol{x}_{\ell,n}, \boldsymbol{z}_\ell\right) + \log \frac{p(\boldsymbol{z}_\ell)}{q_{\boldsymbol{\phi}_\ell}(\boldsymbol{z}_\ell)}\right] + \mathrm{KL}\left[q_{\boldsymbol{\phi}_\ell}(\cdot) \,\|\, p_{\boldsymbol{\theta}}(\cdot \mid \mathcal{D}_\ell)\right], \tag{4.14}$$

where the first term on the right hand side is the ELBO w.r.t. $\log Z_\ell(\boldsymbol{\theta})$, which we denote by $\mathcal{L}(\boldsymbol{\theta})$. We expect $\mathcal{L}(\boldsymbol{\theta})$ to be comparably tight, as our inference scheme allows accurate GMM TP approximations $q_{\boldsymbol{\phi}_\ell}$, s.t., the KL term will be small. Consequently, maximization of $Z_\ell(\boldsymbol{\theta})$ w.r.t. $\boldsymbol{\theta}$ can be performed efficiently by maximization of $\mathcal{L}(\boldsymbol{\theta})$ (I3), cf. also App. B.1.4.1. As is standard, we use the Adam optimizer [Kin15] to perform updates in $\boldsymbol{\theta}$, with MC gradient estimates from samples $\boldsymbol{z}_{\ell,s} \sim q_{\boldsymbol{\phi}_\ell}(\boldsymbol{z}_\ell)$:

$$\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}) \propto \sum_{s,n} \nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}\left(\boldsymbol{y}_{\ell,n} \mid \boldsymbol{x}_{\ell,n}, \boldsymbol{z}_{\ell,s}\right). \tag{4.15}$$

4.4.5 Meta-Training

The goal of any BML algorithm is to compute accurate predictions with well-calibrated uncertainty estimates according to Eq. (4.7), based on samples from the approximate TP $q_{\boldsymbol{\phi}_*}(\boldsymbol{z}_*) \approx p_{\boldsymbol{\theta}}(\boldsymbol{z} \mid \mathcal{D}_*^{\mathrm{c}})$, conditioned on a context set $\mathcal{D}_*^{\mathrm{c}}$ from a target task. During a meta-training stage on meta-data $\mathcal{D}_{1:L}$, we aim to encode inductive biases in the model parameters $\boldsymbol{\theta}$, s.t. small (few-shot) context sets $\mathcal{D}_*^{\mathrm{c}}$ suffice for accurate predictions. To find versatile solutions that work for variable context set sizes, it is necessary to emulate this during meta-training by evaluating gradients for $\boldsymbol{\theta}$ on samples $\boldsymbol{z}_{\ell,s}$ from approximate TPs informed by a range of context set sizes. Standard NPs achieve this by sampling a minibatch of auxiliary subtasks, with a random number of datapoints, from $\mathcal{D}_{1:L}$ for each step in the parameters $\boldsymbol{\Phi}$ (cf. Sec. B.3.2). Our algorithm uses a similar approach: starting from a fixed set of randomly initialized variational GMMs $\boldsymbol{\phi}_\ell$, and a randomly initialized model $\boldsymbol{\theta}$, we iterate through the meta-data in minibatches of auxiliary subtasks, and perform one update step in $\boldsymbol{\phi}_\ell$ for all subtasks in the minibatch, according to Eqs. (4.13), followed by one gradient step in $\boldsymbol{\theta}$. Thus, variational and model parameters evolve jointly in a similar fashion as for standard NP, resulting in a meta-training stage with comparable computational complexity, cf. App. B.5.6. As this approach retains a fixed set of variational GMMs over the whole course of meta-training (one for each auxiliary subtask), we accordingly sample a fixed set of auxiliary subtasks at the beginning of meta-training. We summarize our algorithm in App. B.1, Alg. 1.

4.4.6 Predictions

As our architecture does not amortize inference over tasks and, thus, does not learn a set encoder architecture, the variational GMMs learned during meta-training are not required for predictions on test tasks and can be discarded. To make predictions, we fix the model parameters $\boldsymbol{\theta}$ and fit a new variational GMM $q_{\boldsymbol{\phi}_*}$ to $\mathcal{D}_*^c$ by iterating Eqs. (4.13) until convergence. Afterwards, we can cheaply generate arbitrarily many samples $\boldsymbol{z}_{*,s} \sim q_{\boldsymbol{\phi}_*}(\boldsymbol{z}_*)$, and generate corresponding function samples, evaluated at arbitrary input locations $\boldsymbol{x}_*$, by a single forward pass through the decoder DNN to approximate the predictive distribution according to Eq. (4.7), cf. App. B.1, Alg. 2.

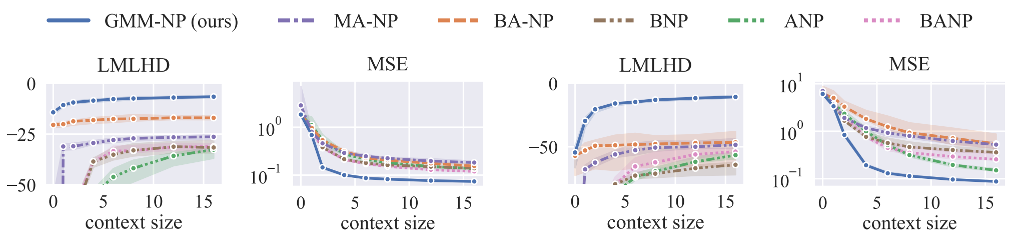
4.5 Empirical Evaluation

Our empirical evaluation aims to study the effect on the predictive performance of (i) our improved TRNG-VI approach as well as of (ii) expressive variational GMM TP approximations in NP-based BML, in (iii) comparison to the state-of-the-art on (iv) a range of practically relevant meta-learning tasks. To this end, we evaluate our GMM-NP architecture on a diverse set of BML experiments, and present comparisons to state-of-the-art BML algorithms, namely the original NP with mean context aggregation (MA-NP) [Gar18c], the NP with Bayesian context aggregation (BA-NP) [Vol21], the Attentive NP (ANP) [Kim19], as well as the Bootstrapping (Attentive) NP (B(A)NP) [Lee20]. Tab. B.1 in App. B.2 gives an overview of the architectural differences of these algorithms. We move details on data generation to App. B.4, and on the baseline implementations to App. B.2. For a fair comparison, we employ a fixed experimental protocol for all datasets and models: we first perform a Bayesian hyperparameter search (HPO) to determine optimal algorithm settings, individually for each model-dataset combination. We then retrain the best model with 8 different random seeds and report the median log marginal predictive likelihood (LMLHD) as well as the median mean squared error (MSE), both in dependence of the context set size. To foster reproducibility, we provide further details on our experimental protocol in App. B.3, the resulting hyperparameters and architecture sizes in App. B.5.7, and publish our source code.[1] Lastly, we include a detailed discussion of limitations and computational resources in App. B.5.6.

4.5.1 Synthetic Datasets

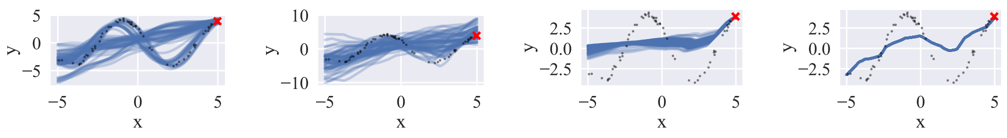
We first study two synthetic function classes [Fin17, Fin18] on which predictions can be easily visualized: (i) sinusoidal functions with varying amplitudes and phases, as well as (ii) a mix of these sinusoidal functions with affine functions with varying slopes and intercepts. Fig. 4.3 shows that our GMM-NP outperforms all baselines by a large margin over the whole range of context sizes, both in terms of LMLHD and MSE.

[1] https://github.com/ALRhub/gmm_np



(a) Sinusoidal functions.

(b) Mix of affine and sinusoidal functions.



(c) GMM-NP (ours).

(d) BA-NP.

(e) ANP.

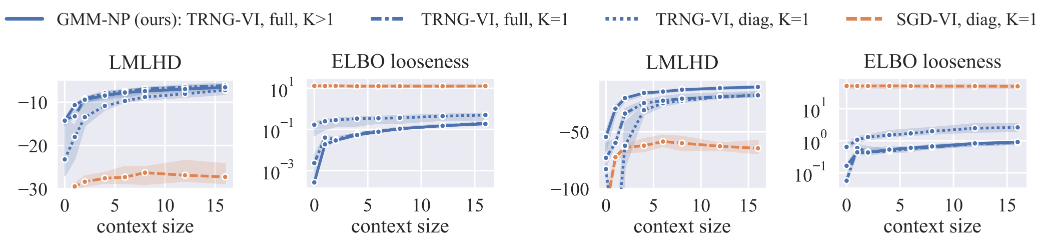
(f) BANP.

Figure 4.3: Panels (a), (b): LMLHD and MSE on two synthetic function classes. Panels (c) – (f): function samples of models trained on the affine-sinusoidal class (b), given one context example (red) from a sinusoidal instance (black). GMM-NP outperforms the baselines, as it accurately quantifies epistemic uncertainty through diverse samples. BA-NP also shows variability in its samples, but does not achieve competitive performance due to its inaccurate TP approximation. ANP and BANP produce essentially deterministic predictions that fail to give reasonable estimates of the predictive distribution. Cf. also Figs. B.5, B.6 in App. B.5.4.

This indicates that GMM-NP's improved TP approximation indeed yields improved epistemic uncertainty estimation (higher LMLHD). Interestingly, GMM-NP also shows improved accuracy (lower MSE) and, notably, achieves this without any additional architectural modifications like parallel deterministic paths with attention modules. This is particularly pleasing, as the results show that such deterministic paths indeed improve accuracy, but degrade epistemic uncertainty estimation massively: (B)ANP performs worst in terms of LMLHD. This is further substantiated by (i) observing that MA-NP and BA-NP, both of which don't employ deterministic paths, are among the best baselines w.r.t. LMLHD, and (ii) by visualizing model predictions (Figs. 4.3, B.5, B.6), demonstrating that (B)ANP compute essentially deterministic function samples that fail to correctly estimate the predictive distribution, while our GMM-NP estimates uncertainty well through variable samples. BNP does not achieve competitive performance, presumably because the bootstrapping approach does not work well for small context sets.

4.5.2 Ablation: Task Posterior Inference

We now demonstrate that GMM-NP's improved performance can indeed be explained by the improved TRNG-VI algorithm with accurate GMM TP approximation. To this end, we compare: (i) BA-NP, i.e., amortized VI with reparameterized gradients and unimodal, factorized Gaussian TP (SGD-VI, diag, $K = 1$), (ii) our GMM-NP, i.e., non-amortized TRNG-VI and full-covariance GMM TP (TRNG-VI, full, $K > 1$), as well as two models employing TRNG-VI, but a unimodal Gaussian TP with (iii) full, and (iv) diagonal covariance. The results are shown in Fig. 4.4.



(a) Sinusoidal functions. **(b)** Mix of affine and sinusoidal functions.

Figure 4.4: LMLHD and ELBO looseness over context set size for different versions of our algorithm (blue). Our improved TRNG-VI inference scheme yields tighter ELBOs than standard SGD-VI (orange) and, thus, improved performance (cf. text and App. B.1.4.1 for details).

In addition, we compare (v) an architecture with full-covariance GMM TP, but trained with iBayes-GMM [Lin20], i.e., with direct step size control instead of trust regions (Fig. B.1, App. B.5.1).

4.5.2.1 VI Algorithm

Considering the LMLHD metric, we observe a significant performance boost when keeping the traditional factorized Gaussian approximation, but switching from SGD-VI to TRNG-VI, indicating that the standard SGD-VI approach is indeed suboptimal for BML. To study this further, we estimate the looseness of the ELBO (cf. App. B.3.3), i.e., the median (over tasks) value of the KL divergence KL $\left[q_{\phi_\ell}(\cdot) \,\|\, p_\theta(\cdot \mid \mathcal{D}_\ell)\right]$ between the true and approximate TPs. We observe that TRNG-VI provides ELBOs that are tighter by at least one order of magnitude in comparison to SGD-VI. As discussed above, this allows for more efficient optimization of the model parameters $\boldsymbol{\theta}$, explaining the performance gain. Lastly, we find that trust regions yield tighter ELBOs than direct step size control and, consequently, improve predictive performance, cf. Fig. B.1, App. B.5.1

4.5.2.2 Posterior Expressivity

We now study the effect of increasing the expressiveness of the TP approximation. This discussion is supplemented by Fig. 4.5, where we visualize the TP and its approximation for a $d_z = 2$ dimensional latent space. First, we observe tighter ELBOs and improved performance when considering full-covariance (but still unimodal) Gaussian TP approximations, and this effect is particularly pronounced for small context sets. This is intuitive, as small context sizes leave a lot of task ambiguity, leading to highly correlated latent dimensions (Fig. 4.5). If we now switch to multimodal TP approximations, i.e., our full GMM-NP architecture with $K > 1$ components (K optimized by HPO), we observe a further increase in performance, as the multimodality of the true TP can be captured more accurately (Figs. 4.5,B.7). This effect is especially pronounced for the affine-sinusoidal mix, but also present for the purely sinusoidal function class. As more complex function classes exhibit stronger task ambiguity, the TP will likely exhibit multimodal, correlated structure over wider ranges of context sizes, s.t. an accurate TP approximation will be even more important.

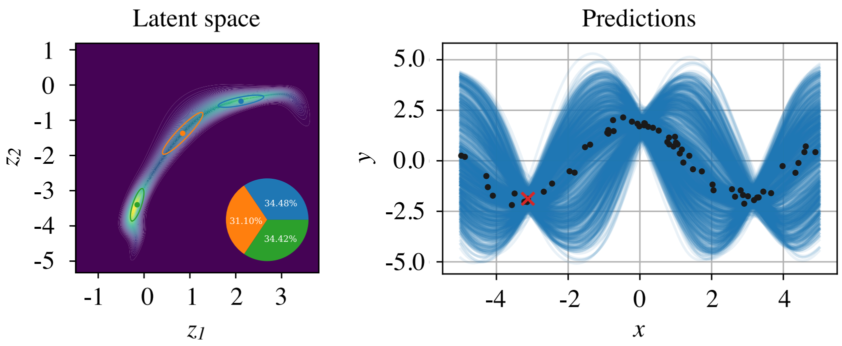
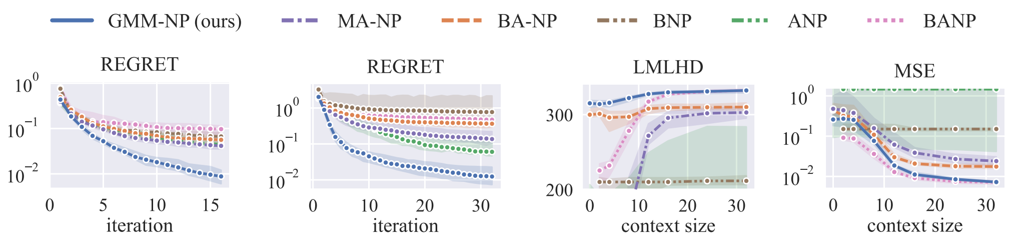


Figure 4.5: Visualization of our GMM-NP model for a $d_z = 2$ dimensional latent space, trained on a meta-dataset of sinusoidal functions with varying amplitudes and phases, after having observed a single context example (red cross, right panel) from an unseen task (black dots, right panel). Left panel: unnormalized task posterior (TP) distribution (contours) and GMM TP approximation with $K = 3$ components (ellipses, mixture weights in %). Right panel: corresponding function samples from our model (blue lines). A single context example leaves much task ambiguity, reflected in a highly correlated, multi-modal TP. Our GMM approximation correctly captures this: predictions are in accordance with (i) the observed data (all samples pass close to the red context example), and with (ii) the learned inductive biases (all samples are sinusoidal), cf. also Fig. B.7 in App. B.5.5

4.5.3 Bayesian Optimization

One important application area for probabilistic regression models is as the surrogate model of Bayesian optimization (BO), a global black-box optimization algorithm well-known for its sampling efficiency [Sha16]. BO serves as an interesting experiment to benchmark Bayesian models, as it relies on well-calibrated uncertainty estimates in order to trade-off exploration against exploitation, which is crucial for efficient optimization. As proposed by Garnelo et al. [Gar18c], we use Thompson sampling [Rus18a] as the BO acquisition function and present results on four function classes: (i) 1D functions sampled from Gaussian process (GP) priors with RBF kernels with varying lengthscales and signal variances [Kim19], and (ii) three function classes [Vol20] obtained by randomly translating and scaling the global optimization benchmark functions Forrester (1D) [For08], Branin (2D) [Pic13], and Hartmann-3 (3D) [Sze78]. In Figs. 4.6a, 4.6b, B.2, we report the median simple regret, i.e., the difference of the current incumbent value to the function's minimum, over BO iteration. We observe that our GMM-NP model represents a more powerful BO surrogate compared to the baselines, providing further evidence that TRNG-VI with GMM TP approximations yields superior epistemic uncertainty estimates. Due to space constraints, we move the results for RBF-GP and 1D Forrester, as well as for the LMLHD and MSE metrics to App. B.5.2, Figs. B.2, B.3.



(a) BO on Branin (2D). **(b)** BO on Hartmann (3D). **(c)** 4D Furuta dynamics prediction.

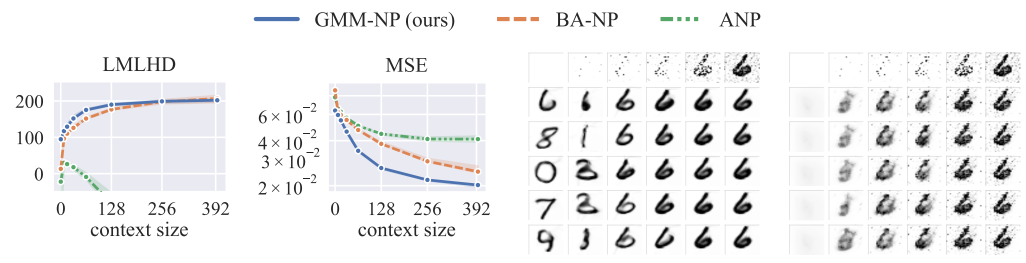
Figure 4.6: Panels (a), (b): simple regret over iteration, when using BML models as Bayesian optimization (BO) surrogates (further results in App. B.5.2). As BO relies on well-calibrated uncertainty predictions, the results demonstrate that GMM-NP provides superior uncertainty estimates. Panel (c): log marginal likelihood (LMLHD) and MSE on one-step ahead predictions of 4D Furuta pendulum dynamics. While GMM-NP generally performs best, BANP also shows strong results.

4.5.4 Dynamics Modeling

We further investigate a challenging dynamics modeling problem on a function class obtained by simulating a Furuta pendulum [Fur92], a highly non-linear 4D dynamical system, as proposed by Volpp et al. [Vol21]. The task is to predict the difference of the next system state $\boldsymbol{x}_{\text{next}} \in \mathbb{R}^4$ to the current system state $\boldsymbol{x} \in \mathbb{R}^4$, i.e., we study one-step ahead dynamics predictions $\boldsymbol{x} \rightarrow \boldsymbol{y} = \Delta\boldsymbol{x} \equiv \boldsymbol{x}_{\text{next}} - \boldsymbol{x} \in \mathbb{R}^4$. The function class is generated by simulating $L = 64$ episodes of $N = 64$ time steps each ($\Delta t = 0.1$ s), where for each episode we randomly sample the 7 physical parameters of the pendulum (3 lengths, 2 masses, 2 friction coefficients). The results (Fig. 4.6c) show that GMM-NP outperforms the baselines in terms of LMLHD by a large margin, demonstrating its applicability to complex dynamics prediction tasks where reliable uncertainty estimates are required, e.g., in robotics applications [Dei11a]. Interestingly, while neither ANP nor BNP can reliably solve this task, BANP performs strongly, reaching GMM-NPs asymptotic performance in terms of LMLHD and yielding even slightly better MSE for small context sets.

4.5.5 Image Completion

To show that our architecture scales to large meta-datasets, we provide results on a 2D image completion experiment on the MNIST database of handwritten digits [LeC10], as proposed by Garnelo et al. [Gar18c]. The task is to predict pixel intensities $y \in \mathbb{R}$ at 2D pixel locations $\boldsymbol{x} \in \mathbb{R}^2$, given a set of context pixels. To obtain a realistic regression task, we add Gaussian noise to each context pixel. The meta-dataset consists of $L = 60000$ images with $N = 784$ pixels each. The results (Fig. 4.7) are consistent with our previous findings: GMM-NP yields markedly improved performance, outperforming the baselines over the whole range of context sizes.



(a) Log marginal likelihood (LMLHD) and MSE. **(b)** GMM-NP (ours). **(c)** ANP.

Figure 4.7: Results on 2D image completion on MNIST. Panels (b), (c) visualize predictions on an unseen task showing the digit "6". The first row shows the context pixels, the remaining rows show five corresponding samples. The results are consistent with earlier observations (e.g., Fig. 4.3): our GMM-NP model shows highly variable samples for small context sets, yielding an accurate estimate of epistemic uncertainty, and contracts properly around the ground truth when more context information is available. ANP yields crisp predictions but massively overfits to the noise, explaining bad LMLHD and MSE scores. We provide further results in App. B.5.3, Fig. B.4.

The architectures with deterministic paths ((B)ANP) fail at properly estimating epistemic uncertainties, leading to low LMLHD values, i.p., for large context sizes. Figs. 4.7b, 4.7c, and B.4 explain why this is the case: GMM-NP (and also, to some extent, BA-NP) generate meaningful images of high variability, corresponding to well-calibrated uncertainty estimates. In contrast, (B)ANP produce essentially deterministic samples that overfit the noise in the context data. While these samples might appear less blurry than those of GMM-NP and BA-NP, they represent inferior solutions of the regression problem.

4.6 Conclusion and Outlook

We proposed GMM-NP, a novel BML algorithm inspired by the NP model architecture. Our approach focuses on accurate task posterior inference, a central algorithmic building block that until now has been treated by amortized inference with set encoders optimized using standard, reparameterized gradients. We demonstrate that this approach leads to suboptimal task posterior approximations and, thus, inefficient optimization of model parameters. We apply modern TRNG-VI techniques that enable expressive variational GMMs, which yields tight ELBOs, efficient optimization, and markedly improved predictive performance in terms of both epistemic uncertainty estimation and accuracy. Despite its simplicity, GMM-NP outperforms the state-of-the-art on a range of experiments and demonstrates its applicability in practical settings, i.p., when meta and context data is scarce. This demonstrates that complex architectural extensions, like Bayesian set encoders or deterministic, attentive computation paths are not required – in fact, we observe that deterministic modules degrade epistemic uncertainty estimation. Therefore, we hope that our work inspires further research on accurate task posterior inference as this turns out to suffice for accurate BML.

5 Meta-Learning Acquisition Functions for Bayesian Optimization

In Secs. 3 and 4, we developed improved task posterior inference schemes for NP-based BML models and showed that these methods yield powerful surrogate models for multitask Bayesian optimization (BO) (Sec. 4.5.3). Recall from Sec. 2.1.7 that in addition to the Bayesian surrogate model, BO requires the specification of an optimization strategy based on the probabilistic predictions of the surrogate model in the form of an acquisition function (AF). This raises the interesting question of whether meta-learning optimization could also be performed at the AF level instead of the surrogate model level.

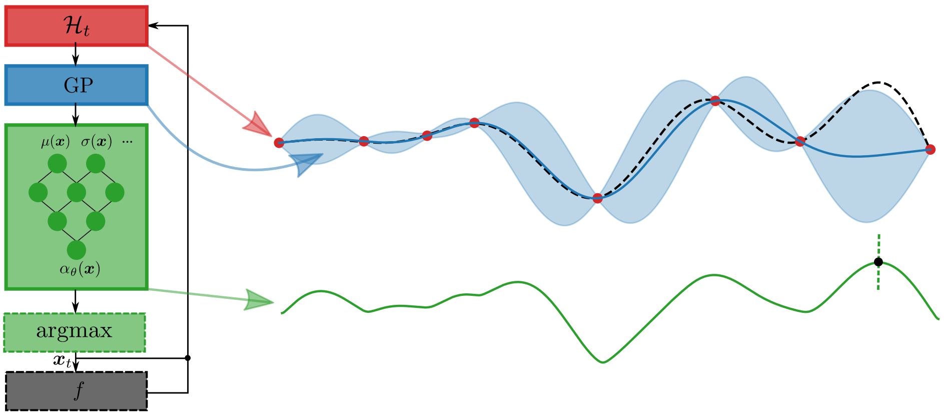

$\mathcal{H}_t$
GP
$\mu(\boldsymbol{x})$ $\sigma(\boldsymbol{x})$ ...
$\alpha_\theta(\boldsymbol{x})$
argmax
$\boldsymbol{x}_t$
f

Figure 5.1: Visualization of our MetaBO algorithm for meta-learning BO. We keep the standard BO loop (Fig. 2.2) and only replace the AF (green) with a NN operating on the predictions of a GP surrogate model (blue) that fits the optimization history $\mathcal{H}_t$ (red) and computes Bayesian predictions $\mu(\boldsymbol{x}) \equiv \mathbb{E}_{p(\boldsymbol{y}_t|\boldsymbol{x},\mathcal{H}_t)}[\boldsymbol{y}_t]$ and $\sigma^2(\boldsymbol{x}) \equiv \mathbb{V}_{p(\boldsymbol{y}_t|\boldsymbol{x},\mathcal{H}_t)}[\boldsymbol{y}_t]$. We train the neural AF $\alpha_\theta(\boldsymbol{x})$ to maximize optimization performance on a meta-dataset of similar optimization problems. In this way, we inject inductive biases towards this class of problems into the optimization algorithm, which allows unseen problem instances to be solved in a data-efficient way.

In this chapter, we explore this idea and develop MetaBO, a novel algorithm for multitask BO that injects learned inductive biases towards a meta-dataset of similar optimization tasks into the AF (Fig. 5.1). This AF is parameterized by a NN that operates on the predictions of a standard single-task surrogate model, such as a Gaussian process (GP). Therefore, it can serve as a plug-in replacement for standard AFs in any BO framework. While the NP-based BML algorithms studied previously are optimized for predictive performance, MetaBO optimizes its neural AF directly for optimization performance using reinforcement learning (RL) [Sut18]. This results in a bi-level optimization scheme, where the RL algorithm is the outer learner and the neural AF is the base learner, so that MetaBO can be regarded as an instance of optimization-based meta-learning (Sec. 2.2.5). We demonstrate that the resulting algorithm yields strong performance on a number of challenging meta-optimization tasks.

The remainder of this chapter has been published as [Vol20] Volpp, Michael; Fröhlich, Lukas P.; Fischer, Kirsten; Doerr, Andreas; Falkner, Stefan; Hutter, Frank, and Daniel, Christian: "Meta-Learning Acquisition Functions for Transfer Learning in Bayesian Optimization". In: International Conference on Learning Representations *(2020). Reprinted with permission from the authors.*

5.1 Introduction

Global optimization of black-box functions is highly relevant for a wide range of real-world tasks. Examples include the tuning of hyperparameters in machine learning, the identification of control parameters, or the optimization of system designs. Such applications oftentimes require the optimization of relatively low-dimensional ($\lesssim 10D$) functions where each function evaluation is expensive in either time or cost. Furthermore, there is typically no gradient information available.

In this context of data-efficient global black-box optimization, Bayesian optimization (BO) has emerged as a powerful solution [Moc75, Bro10, Sno12, Sha16]. BO's data efficiency originates from a probabilistic surrogate model which is used to generalize over information from individual data points. This model is typically given by a Gaussian process (GP), whose well-calibrated uncertainty prediction allows for an informed exploration-exploitation trade-off during optimization. The exact manner of performing this trade-off, however, is left to be encoded in an acquisition function (AF). There is a wide range of AFs available in the literature which are designed to yield universal optimization strategies and therefore come with minimal assumptions about the class of target objective functions.

To achieve optimal data-efficiency on new instances of previously seen tasks, however, it is crucial to incorporate the information obtained from these tasks into the optimization. Therefore, transfer learning is an important and active field of research. Indeed, in many practical applications, optimizations are repeated numerous times in similar settings, underlining the need for specialized optimizers. Examples include hyperparameter optimization which is repeatedly done for the same machine learning model on varying datasets or the optimization of control parameters for a given system with varying physical configurations.

Following recent approaches [Swe13, Feu18, Wis18], we argue that it is beneficial to perform transfer learning for global black-box optimization in the framework of BO to retain the proven generalization capabilities of its underlying GP surrogate model. To not restrict the expressivity of this model, we propose to implicitly encode the task structure in a specialized AF, i.e., in the optimization strategy. We realize this encoding via a novel method which meta-learns a neural

AF, i.e., a neural network representing the AF, on a set of source tasks. The meta-training is performed using reinforcement learning, making the proposed approach applicable to the standard BO setting, where we do not assume access to objective function gradients.

Our contributions are (1) a novel transfer learning method allowing the incorporation of implicit structural knowledge about a class of objective functions into the framework of BO through learned neural AFs to increase data-efficiency on new task instances, (2) an automatic and practical meta-learning procedure for training such neural AFs which is fully compatible with the black-box optimization setting, i.e, not requiring objective function gradients, and (3) the demonstration of the efficiency and practical applicability of our approach on a challenging simulation-to-real control task, on two hyperparameter optimization problems, as well as on a set of synthetic functions.

5.2 Related Work

The general idea of improving the performance or convergence speed of a learning system on a given set of tasks through experience on similar tasks is known as learning to learn, meta-learning or transfer learning and has attracted a large amount of interest in the past while remaining an active field of research [Sch87, Hoc01, Thr98, Lak16].

In the context of meta-learning optimization, a large body of literature revolves around learning local optimization strategies. One line of work focuses on learning improved optimizers for the training of neural networks, e.g., by directly learning update rules [Ben91, Run00] or by learning controllers for selecting appropriate step sizes for gradient descent [Dan16]. Another direction of research considers the more general setting of replacing the gradient descent update step by neural networks which are trained using either reinforcement learning [Li16, Li17a] or in a supervised fashion [And16, Met19]. Finn et al. [Fin17], Nichol et al. [Nic18], and Flennerhag et al. [Fle19] propose approaches for initializing machine learning models through meta-learning to be able to solve new learning tasks with few gradient steps.

We are currently aware of only one work tackling the problem of meta-learning global black-box optimization [Che17]. In contrast to our proposed method, the authors assume access to gradient information and choose a supervised learning approach, representing the optimizer as a recurrent neural network operating on the raw input vectors. Based on statistics of the optimization history accumulated in its memory state, this network directly outputs the next query point. In contrast, we consider transfer learning applications where gradients are typically not available.

A number of articles address the problem of increasing BO's data-efficiency via transfer learning, i.e., by incorporating information obtained from similar optimizations on source tasks into the current target task. A range of methods accumulate all available source and target data in a single GP and make the data comparable via a ranking algorithm [Bar13], standardization or multi-kernel GPs [Yog14], multi-task GPs [Swe13], the GP noise model [The16], or by regressing on prediction biases [Shi17]. These approaches naturally suffer from the cubic scaling behaviour of GPs, which can be tackled for instance by replacing the GP model, e.g., with Bayesian neural networks with task-specific embedding vectors [Spr16] or with adaptive Bayesian linear regression with basis functions shared across tasks via a neural network [Per18]. Recently, Garnelo et al. [Gar18c] proposed Neural Processes as another interesting alternative for GPs with improved scaling behavior. Other approaches retain the GP surrogate model and combine individual GPs for source and target tasks in an ensemble model with the weights adjusted according to the GP uncertainties [Sch16a], dataset similarities [Wis16], or estimates of the GP generalization performance on the target task [Feu18]. Similarly, Golovin et al. [Gol17] form a stack of GPs by iteratively regressing onto the residuals w.r.t. the most recent source task. In contrast to our proposed method, many of these approaches rely on hand-engineered dataset features to measure the relevance of source data for the target task. Such features have also been used to pick promising initial configurations for BO [Feu15a, Feu15b].

The method being closest in spirit and capability to our approach is proposed by Wistuba et al. [Wis18]. It is similar to the aforementioned ensemble techniques with the important difference that the source and target GPs are not combined via a surrogate model but via a new AF, the so-called transfer acquisition function (TAF). This AF is defined to be a weighted superposition of the predicted improvements according to the source GPs and the expected improvement according to the target GP. Viewed in this context, our method also combines knowledge from source and target tasks in a new AF which we represent as a neural network. Our weighting of source and target data is implicitly determined in a meta-learning phase and is automatically regulated during the optimization on the target task to adapt online to the specific objective function at hand. Furthermore, our method does not store and evaluate many source GPs because the knowledge from the source datasets is encoded directly in the network weights of the learned AF. This allows our method to incorporate large amounts of source data while the applicability of TAF is restricted to a comparably small number of source tasks.

5.3 Preliminaries

We are aiming to find a global optimum $x^* \in \arg\max_{x \in \mathcal{D}} f(x)$ of some unknown objective function $f : \mathcal{D} \to \mathbb{R}$ on the domain $\mathcal{D} \subset \mathbb{R}^D$. The only means of acquiring information about f is via (possibly noisy) evaluations at points in $\mathcal{D}$. Therefore, at each optimization step $t \in \{1, 2, \ldots\}$, the optimizer has to decide for the iterate $x_t \in \mathcal{D}$ solely based on the *optimization history* $\mathcal{H}_t \equiv \{x_i, y_i\}_{i=1}^{t-1}$ with $y_i = f(x_i) + \epsilon$. Here, $\epsilon \sim \mathcal{N}\left(0, \sigma_n^2\right)$ denotes independent and identically distributed Gaussian noise. In particular, the optimizer does not have access to gradients of f. To assess the performance of global optimization algorithms, it is natural to use the *simple regret* $R_t \equiv f(x^*) - f(x_t^+)$ where x_t^+ is the input location corresponding to the best evaluation found by an algorithm up to and including step t. The proposed method relies on the framework of BO and is trained using reinforcement learning. Therefore, we now shortly introduce these frameworks.

5.3.1 Bayesian Optimization

In Bayesian optimization (BO) [Sha16], one specifies a prior belief about the objective function f and at each step t builds a probabilistic surrogate model conditioned on the current optimization history $\mathcal{H}_t$. Typically, a Gaussian process (GP) [Ras05] is employed as the surrogate model in which case the resulting posterior belief about $f(x)$ follows a Gaussian distribution with mean $\mu_t(x) \equiv \mathbb{E}\{f(x) \,|\, \mathcal{H}_t\}$ and variance $\sigma_t^2(x) \equiv \mathbb{V}\{f(x) \mid \mathcal{H}_t\}$, for which closed-form expressions are available. To determine the next iterate x_t based on the belief about f given $\mathcal{H}_t$, a sampling strategy is defined in terms of an *acquisition function* (AF) $\alpha_t(\cdot \mid \mathcal{H}_t) : \mathcal{D} \to \mathbb{R}$. The AF outputs a score value at each point in $\mathcal{D}$ such that the next iterate is defined to be given by $x_t \in \arg\max_{x \in \mathcal{D}} \alpha_t(x \mid \mathcal{H}_t)$. The strength of the resulting optimizer is largely based upon carefully designing the AF to trade-off exploration of unknown versus exploitation of promising areas in $\mathcal{D}$.

There is a wide range of general-purpose AFs available in the literature. Popular choices are *probability of improvement* (PI) [Kus64], *GP-upper confidence bound* (GP-UCB) [Sri10], and *expected improvement* (EI) [Moc75]. In our experiments, we will use EI as a not pre-informed baseline AF, so we state its definition here,

$$\mathrm{EI}_t(x) \equiv \mathbb{E}_{f(x)}\left\{\max\left[f(x) - f(x_{t-1}^+), 0\right] \mid \mathcal{H}_t\right\}, \tag{5.1}$$

and note that it can be written in closed form if $f(x)$ follows a Gaussian distribution.

To perform transfer learning in the context of BO, Wistuba et al. [Wis18] introduced the *transfer acquisition framework* (TAF) which defines a new AF as a weighted superposition of EI on the target task and the predicted improvements on the source tasks, i.e.,

$$\mathrm{TAF}_t(x) \equiv \frac{w_{M+1}\mathrm{EI}_t^{M+1}(x) + \sum_{j=1}^{M} w_j I_t^j(x)}{\sum_{j=1}^{M+1} w_j}, \tag{5.2}$$

with the predicted improvement

$$I_t^j(x) \equiv \max\left(\mu^j(x) - y_{t-1}^{j,\max}, 0\right). \tag{5.3}$$

TAF stores separate GP surrogate models for the source and target tasks, with $j \in \{1, \ldots, M\}$ indexing the source tasks and $j = M + 1$ indexing the target task. Therefore, EI_t^{M+1} denotes EI according to the target GP surrogate model and μ^j denotes the mean function of the j-th source GP model. $y_t^{j,\max}$ denotes the maximum of the mean predictions of the j-th source GP model on the set of iterates $\{x_i\}_{i=1}^t$. The weights $w_j \in \mathbb{R}$ are determined either based on the predicted variances of the source and target GP surrogate models (TAF-ME) or, alternatively, by a pairwise comparison of the predicted performance ranks of the iterates (TAF-R).

5.3.2 Reinforcement Learning

Reinforcement learning (RL) allows an agent to learn goal-oriented behavior via trial-and-error interactions with its environment [Sut18]. This interaction process is formalized as a Markov decision process: at step t the agent senses the environment's state $s_t \in \mathcal{S}$ and uses a policy $\pi : \mathcal{S} \to \mathcal{P}(\mathcal{A})$ to determine the next action $a_t \in \mathcal{A}$. Typically, the agent explores the environment by means of a probabilistic policy, i.e., $\mathcal{P}(\mathcal{A})$ denotes the probability measures over $\mathcal{A}$. The environment's response to a_t is the next state s_{t+1}, which is drawn from a probability distribution with density $p(s_{t+1} \mid s_t, a_t)$. The agent's goal is formulated in terms of a scalar reward $r_t = r(s_t, a_t, s_{t+1})$, which the agent receives together with s_{t+1}. The agent aims to maximize the expected cumulative discounted future reward $\eta(\pi)$ when acting according to π and starting from some state $s_0 \in \mathcal{S}$, i.e., $\eta(\pi) \equiv \mathbb{E}_\pi\left[\sum_{t=1}^{T} \gamma^{t-1} r_t \mid s_0\right]$. Here, T denotes the episode length and $\gamma \in (0, 1]$ is a discount factor.

5.4 MetaBO Algorithm

5.4.1 Neural Acquisition Functions

We devise a global black-box optimization method that is able to automatically identify and exploit structural properties of a given class of objective functions for improved data-efficiency. We stay within the framework of BO, enabling us to exploit the powerful generalization capabilities of a GP surrogate model. The actual optimization strategy which is informed by this GP is classically encoded in a hand-designed AF. Instead, we meta-train on a set of source tasks to replace this AF by a neural network but retain all other elements of the proven BO-loop (middle panel of Fig. 5.2). To distinguish the learned AF from a classical AF α_t, we call such a network a *neural acquisition function* and denote it by $\alpha_{t,\theta}$, indicating that it is parameterized by a vector θ. We dub the resulting algorithm *MetaBO*.

Let $\mathcal{F}$ be the class of objective functions for which we aim to learn a neural acquisition function $\alpha_{t,\theta}$. For instance, $\mathcal{F}$ may be the set of objective functions resulting from different physical configurations of a laboratory experiment or from evaluating the loss function of a machine learning model on different data sets. Often, such objective functions share structure which we aim to exploit for data-efficient optimization on further instances from the same function class. In many relevant cases, it is straightforward to obtain approximations to $\mathcal{F}$, i.e., a set of functions $\mathcal{F}'$ which capture relevant properties of $\mathcal{F}$ but are much cheaper to evaluate (e.g., by using numerical simulations or results from previous hyperparameter optimization tasks [Wis18]). During an offline meta-training phase, MetaBO makes use of such cheap approximations to identify the implicit structure of $\mathcal{F}$ and to adapt θ to obtain a data-efficient optimization strategy customized to $\mathcal{F}$.

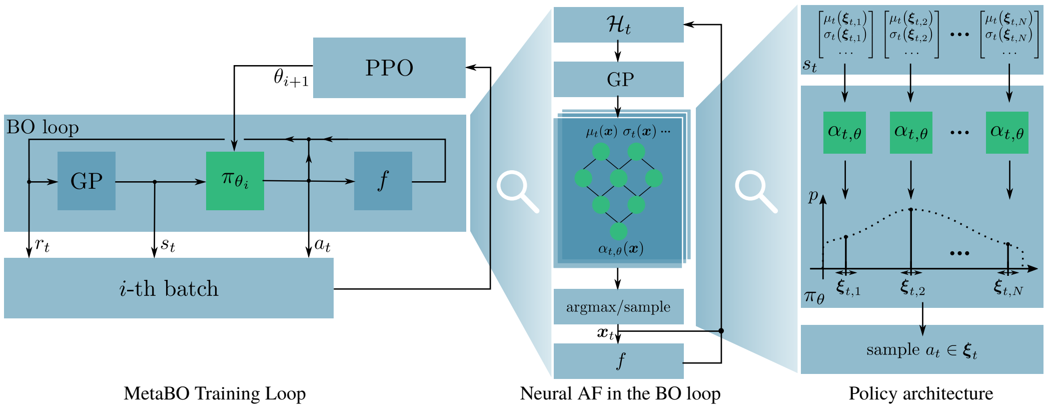


Figure 5.2: Different levels of the MetaBO framework. Left panel: structure of the training loop for meta-learning neural AFs using RL (PPO). Middle panel: the classical BO loop with a neural AF $\alpha_{t,\theta}$. At test time, there is no difference to classical BO, i.e., x_t is given by the arg max of the AF output. During training, the AF corresponds to the RL policy evaluated on an adaptive set $\xi_t \subset \mathcal{D}$. The outputs are interpreted as logits of a categorical distribution from which the actions $a_t = x_t \in \xi_t$ are sampled. This sampling procedure is detailed in the right panel. We indicate by the dotted curve and tiny two-headed arrows that $\alpha_{t,\theta}$ is a function defined on the whole domain $\mathcal{D}$ which can be evaluated at arbitrary points $\xi_{t,n}$ to form the categorical distribution representing the policy π_θ.

Typically, the minimal set of inputs to AFs in BO is given by the pointwise GP posterior prediction $\mu_t(x)$ and $\sigma_t(x)$. To perform transfer learning, the AF has to be able to identify relevant structure shared by the objective functions in $\mathcal{F}$. In our setting, this is achieved via extending this basic set of inputs by additional features which enable the neural AF to evaluate sample locations. Therefore, in addition to the mean $\mu_t(x)$ and variance $\sigma_t(x)$ at potential sample locations, the neural AF also receives the input location x itself. Furthermore, we add to the set of input features the current optimization step t and the optimization budget T, as these features can be valuable for adjusting the exploration-exploitation trade-off [Sri10]. Therefore, we define

$$\alpha_{t,\theta}(x) \equiv \alpha_{t,\theta}[\mu_t(x), \sigma_t(x), x, t, T]. \tag{5.4}$$

This architecture allows learning a scalable neural AF, as we still base our architecture only on the pointwise GP posterior prediction. Furthermore, neural AFs of this form can be used as a plug-in feature in any state-of-the-art BO framework. In particular, if differentiable activation functions are chosen, a neural AF constitutes a differentiable mapping $\mathcal{D} \to \mathbb{R}$ and standard gradient-based optimization strategies can be used to find its maximum in the BO loop during evaluation. We further emphasize that after the training phase the resulting neural AF is fully defined, i.e., there is no need to calibrate any AF-related hyperparameters.

5.4.2 Training Procedure

In the general BO setting, gradients of $\mathcal{F}$ are assumed to be unavailable. This is oftentimes also true for the functions in $\mathcal{F}'$, for instance, when $\mathcal{F}'$ comprises numerical simulations or results from previous optimization runs. Therefore, we resort to RL as the meta-algorithm, as it does not require gradients of the objective functions. Specifically, we use the Proximal Policy Optimization (PPO) algorithm as proposed in Schulman et al. [Sch17]. Tab. 5.1 translates the MetaBO-setting into RL parlance.

Table 5.1: The MetaBO setting in the RL framework.

RL	**MetaBO**
Policy π_θ	Neural AF $\alpha_{t,\theta}$
Episode	Optimization run on $f \in \mathcal{F}'$
Episode length T	Optimization budget T
State s_t	$\left[\mu_t\left(\xi_{t,n}\right), \sigma_t\left(\xi_{t,n}\right), \xi_{t,n}, t, T\right]_{n=1}^{N}$
Action a_t	Sampling point $x_t \in \xi_t$
Reward r_t	Negative simple regret $-R_t$
Transition $p(s_{t+1} \mid s_t, a_t)$	Noisy evaluation of f, GP update

We aim to shape the mapping $\alpha_{t,\theta}(x)$ during meta-training in such a way that its maximum location corresponds to a promising sampling location x for optimization. The meta-algorithm PPO explores its state space using a parameterized stochastic policy π_θ from which the actions $a_t = x_t$ are sampled depending on the current state s_t, i.e., $a_t \sim \pi_\theta(\cdot \mid s_t)$. As the meta-algorithm requires access to the global information contained in the GP posterior prediction, the state s_t at optimization step t *formally* corresponds to the *functions* μ_t and σ_t (together with the aforementioned additional input features to the neural AF). To connect the neural AF $\alpha_{t,\theta}$ with the policy π_θ and to arrive at a practical implementation, we evaluate μ_t and σ_t on a discrete set of points $\xi_t \equiv \left\{\xi_{t,n}\right\}_{n=1}^{N} \subset \mathcal{D}$ and feed these evaluations through the neural AF $\alpha_{t,\theta}$ one at a time, yielding one scalar output value $\alpha_{t,\theta}\left(\xi_{t,n}\right) = \alpha_{t,\theta}\left[\mu_t\left(\xi_{t,n}\right), \sigma_t\left(\xi_{t,n}\right), \xi_{t,n}, t, T\right]$ for each point $\xi_{t,n}$. These outputs are interpreted as the logits of a categorical distribution, i.e., we arrive at the policy architecture

$$\pi_\theta(\cdot \mid s_t) \equiv \operatorname{Cat}\left[\alpha_{t,\theta}\left(\xi_{t,1}\right), \ldots, \alpha_{t,\theta}\left(\xi_{t,N}\right)\right], \tag{5.5}$$

cf. Fig. 5.2, right panel. Therefore, the proposed policy evaluates the *same* neural acquisition function $\alpha_{t,\theta}$ at arbitrarily many input locations $\xi_{t,n}$ and preferably samples actions $x_t \in \xi_t$ with high $\alpha_{t,\theta}(x_t)$. This incentivizes the meta-algorithm to adjust θ such that promising locations $\xi_{t,n}$ are attributed high values of $\alpha_{t,\theta}\left(\xi_{t,n}\right)$.

Calculating a sufficiently fine *static* set ξ of evaluation points is challenging for higher dimensional settings. Instead, we build on the approach proposed by Snoek et al. [Sno12] and continuously adapt $\xi = \xi_t$ to the current state of $\alpha_{t,\theta}$. At each step t, $\alpha_{t,\theta}$ is first evaluated on a static and relatively coarse Sobol grid [Sob67] ξ_{global} spanning the whole domain $\mathcal{D}$. Subsequently, local maximizations of $\alpha_{t,\theta}$ are started from the k points corresponding to the best evaluations. We denote the resulting set of local maxima by $\xi_{\text{local},t}$. Finally, we define $\xi_t \equiv \xi_{\text{local},t} \cup \xi_{\text{global}}$. The adaptive local part of this set enables the RL agent to exploit what it has learned so far by picking points which look promising according to the current neural AF while the static global part maintains exploration. We refer the reader to App. C.2.1 for details.

The final characteristics of the neural AF are controlled through the choice of reward function. For the presented experiments we emphasized fast convergence to the optimum by using the negative simple regret as the reward signal, i.e., we set $r_t \equiv -R_t$.[1] This choice does not penalize explorative evaluations which do not yield an immediate improvement and additionally serves as a normalization of the functions $f \in \mathcal{F}'$. We emphasize that the knowledge of the true maximum is only required during training and that cases in which it is not known at training time do not limit the applicability of our method, as a cheap approximation (e.g., by evaluating the function on a coarse grid) can also be utilized.

The left panel of Fig. 5.2 depicts the resulting training loop graphically. The outer loop corresponds to the RL meta-training iterations, each performing a policy update step $\pi_{\theta_i} \rightarrow \pi_{\theta_{i+1}}$. To approximate the gradients of the PPO loss function, we record a batch of episodes in the inner loop, i.e., a set of (s_t, a_t, r_t)-tuples, by rolling out the current policy π_{θ_i}. At the beginning of each episode, we draw some function f from the training set $\mathcal{F}'$ and fix an optimization budget T. In each iteration of the inner loop we determine the adaptive set ξ_t and feed the state s_t through the policy which yields the action $a_t = x_t$. We then evaluate f at x_t and use the result to compute the reward r_t and to update the optimization history: $\mathcal{H}_t \rightarrow \mathcal{H}_{t+1} = \mathcal{H}_t \cup \{x_t, y_t\}$. Finally, the GP is conditioned on the updated optimization history $\mathcal{H}_{t+1}$ to obtain the next state s_{t+1}.

5.5 Empirical Evaluation

We trained MetaBO on a wide range of function classes and compared the performance of the resulting neural AFs with the general-purpose AF expected improvement (EI)[2] as well as the transfer acquisition function framework (TAF) which proved to be the current state-of-the-art solution for transfer learning in BO in an extensive experimental study [Wis18]. We tested both the ranking-based version (TAF-R) and the mixture-of-experts version (TAF-ME). We refer the reader to App. C.1 for a more detailed experimental investigation of MetaBO's performance.

If not stated differently, we report performance in terms of the median simple regret R_t over 100 optimization runs on unseen test functions as a function of the optimization step t together with 30%/70% percentiles (shaded areas). We emphasize that all experiments use the same MetaBO hyperparameters, making our method easily applicable in practice. Furthermore, MetaBO does not increase evaluation time considerably compared to standard AFs, cf. App. C.1.2, Tab. C.2. In addition, even the most expensive of our experiments (the simulation-to-real task, due to the

[1] Alternatively, a logarithmically-transformed version of this reward signal, $r_t \equiv -\log_{10} R_t$, can be used in situations where high-accuracy solutions shall be rewarded.

[2] We also evaluated probability of improvement (PI) as well as GP-upper confidence bound (GP-UCB) but do not present the results here to avoid clutter, as EI performed better in all our experiments.

simulation in the BO loop) required not more than 10h of training time on a moderately complex architecture (10 CPU workers, 1 GPU), which is fully justified for our intended offline transfer learning use-case. To foster reproducibility, we provide a detailed exposition of the experimental settings in App. C.2 and make the source code of MetaBO available online.[1]

5.5.1 Global Optimization Benchmark Functions

We evaluated our method on a set of synthetic function classes based on the standard global optimization benchmark functions Branin ($D = 2$), Goldstein-Price ($D = 2$), and Hartmann-3 ($D = 3$) [Pic13]. To construct the training set $\mathcal{F}'$, we applied translations in $[-0.1, 0.1]^D$ as well as scalings in $[0.9, 1.1]$.

As TAF stores and evaluates one source GP for each source task, its applicability is restricted to a relatively small amount of source data. For the evaluations of TAF and MetaBO, we therefore picked a random set of $M = 50$ source tasks from the continuously parameterized family $\mathcal{F}'$ of available objective functions and spread these tasks uniformly over the whole range of translations and scalings (MetaBO-50, TAF-R-50, TAF-ME-50). We used $N_{\text{TAF}} = 100$ data points for each source GP of TAF. We also tested both flavors of TAF for $M = 20$ source tasks (with $N_{\text{TAF}} = 50$) and observed that TAF's performance does not necessarily increase with more source data, rendering the choice of suitable source tasks cumbersome. Fig. 5.3 shows the performance on unseen functions drawn randomly from $\mathcal{F}'$.

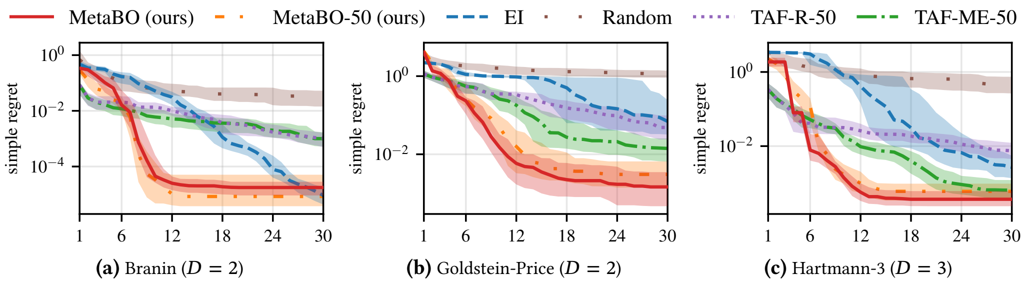


Figure 5.3: Performance on three global optimization benchmark functions with random translations sampled uniformly from $[-0.1, 0.1]^D$ and scalings from $[0.9, 1.1]$. To test TAF's performance, we randomly picked $M = 50$ source tasks from this function class and evaluated both the ranking-based version (TAF-R-50) and the mixture-of-experts version (TAF-ME-50). We trained MetaBO on the same set of source tasks (MetaBO-50). In contrast to TAF, MetaBO can also be trained without manually restricting the set of available source tasks. The corresponding results are labelled "MetaBO". MetaBO outperformed EI by clear margin, especially in early stages of the optimization. After few steps used to identify the specific instance of the objective function, MetaBO also outperformed both flavors of TAF over wide ranges of the optimization budget. Results for TAF-20 can be found in App. C.1.4, Fig. C.7.

To avoid clutter, we move the results for TAF-20 to App. C.1.4, cf. Fig. C.7. MetaBO-50 outperformed EI by large margin, in particular at early stages of the optimization, by making use of the

[1] https://github.com/boschresearch/MetaBO

structural knowledge about $\mathcal{F}'$ acquired during the meta-learning phase. Furthermore, MetaBO-50 outperformed both flavors of TAF-50 over wide ranges of the optimization budget. This is due to its ability to learn sampling strategies which go beyond a combination of a prior over $\mathcal{D}$ and a standard AF (as is the case for TAF). Indeed, note that MetaBO spends some initial non-greedy evaluations to identify specific properties of the target objective function, resulting in much more efficient optimization strategies. We investigate this behaviour further on simple toy experiments and using easily interpretable baseline AFs in App. C.1.1.

We further emphasize that MetaBO does not require the user to manually pick a suitable set of source tasks but that it can naturally learn from the whole set $\mathcal{F}'$ of available source tasks by randomly picking a new task from $\mathcal{F}'$ at the beginning of each BO iteration and aggregating this information in the neural AF weights. We also trained this full version of MetaBO (labelled "MetaBO") on the global optimization benchmark functions, obtaining performance comparable with MetaBO-50. We demonstrate below that for more complex experiments, such as the simulation-to-real task, MetaBO's ability to learn from the full set of available source tasks is crucial for efficient transfer learning. We also investigate the dependence of MetaBO's performance on the number of source tasks in more detail in App. C.1.2.

As a final test on synthetic functions, we evaluated the neural AFs on objective functions outside the training distribution. This can give interesting insights into the nature of the problems under consideration. We move the results of this experiment to App. C.1.3.

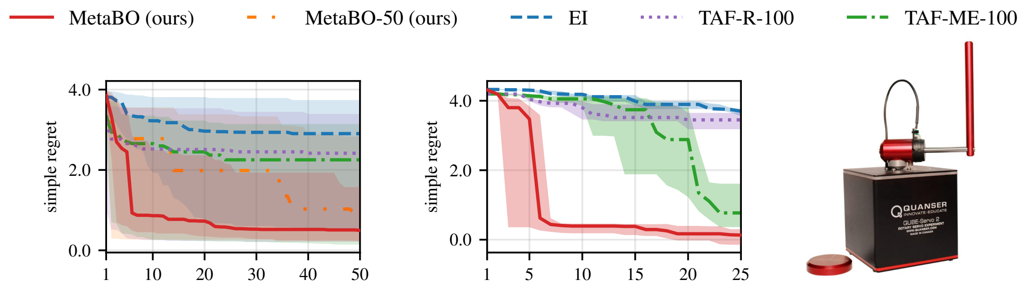
5.5.2 Simulation-to-Real Task

Sample efficiency is of special interest for the optimization of real world systems. In cases where an approximate model of the system can be simulated, the proposed approach can be used to improve the data-efficiency on the real system. To demonstrate this, we evaluated MetaBO on a $4D$ simulation-to-real experiment. The task was to stabilize a Furuta pendulum [Fur92] for 5 s around the upper equilibrium position using a linear state-feedback controller. We applied BO to tune the four feedback gains of this controller [Frö20]. To assess the performance of a given controller, we employed a logarithmic quadratic cost function [Ban17]. If the controller was not able to stabilize the system or if the voltage applied to the motor exceeded some safety limit, we added a penalty term proportional to the remaining time the pendulum would have had to be stabilized for successfully completing the task. We emphasize that the cost function is rather sensitive to the control gains, resulting in a challenging black-box optimization problem.

To meta-learn the neural AF, we employed a fast numerical simulation based on the nonlinear dynamics equations of the Furuta pendulum which only contained the most basic physical effects. In particular, effects like friction and stiction were not modeled. The training distribution was generated by sampling the physical parameters of this simulation (two lengths, two masses), uniformly on a range of 75% − 125% around the measured parameters of the hardware (Quanser

QUBE – Servo 2,[1] Fig. 5.4c). We also used this simulation to generate $M = 100$ source tasks for TAF ($N_{\mathrm{TAF}} = 200$).

Fig. 5.4a shows the performance on objective functions from simulation. Again, MetaBO learned a sophisticated sampling strategy which first identifies the target objective function and adapts its optimization strategy accordingly, resulting in very strong optimization performance. In contrast, TAF's superposition of a prior over $\mathcal{D}$ obtained from the source tasks with EI on the target task leads to excessive explorative behaviour. We move further experimental results for TAF-50 to App. C.1.4, Fig. C.8.



(a) Evaluation in simulation. **(b)** Evaluation on hardware in (c). **(c)** Exp. setup.[3]

Figure 5.4: Performance on a simulation-to-real task (cf. text). MetaBO and TAF used source data from a cheap numerical simulation. (a) Performance on an extended training set in simulation. (b) Transfer to the hardware depicted in (c), averaged over ten BO runs. MetaBO learned robust neural AFs with very strong optimization performance and online adaption to the target objectives, which reliably yielded stabilizing controllers after less than ten BO iterations while TAF-ME-100, TAF-R-100, and EI explore too heavily. Comparing the results for MetaBO and MetaBO-50 in simulation, we observe that MetaBO benefits from its ability to learn from the whole set of available source data, while TAF's applicability is restricted to a comparably small number of source tasks. We move the results for TAF-50 to App. C.1.4, Fig. C.8.

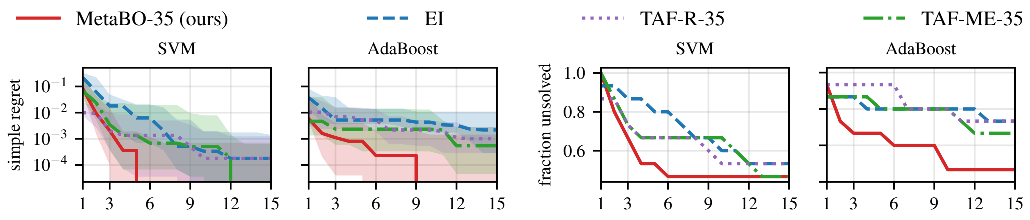
By comparing the performance of MetaBO and MetaBO-50 in simulation, we find that our architecture's ability to incorporate large amounts of source data is indeed beneficial on this complex optimization problem. The results in App. C.1.2 underline that this task indeed requires large amounts of source data to be solved efficiently. This is substantiated by the results on the hardware, on which we evaluated the full version of MetaBO and the baseline AFs obtained by training on data from simulation without any changes. Fig. 5.4b shows that MetaBO learned a neural AF which generalizes well from the simulated objectives to the hardware task and was thereby able to rapidly adjust to its specific properties. This resulted in very data-efficient optimization on the target system, consistently yielding stabilizing controllers after less than ten BO iterations. In comparison, the benchmark AFs required many samples to identify promising regions of the search space and therefore did not reliably find stabilizing controllers within the budget of 25 optimization steps.

[1] https://www.quanser.com/products/qube-servo-2

As it provides interesting insights into the nature of the studied problem, we investigate MetaBO's generalization performance to functions outside the training distribution in App. C.1.3. We emphasize, however, that the intended use case of our method is on unseen functions drawn from the training distribution. Indeed, by measuring the physical parameters of the hardware system and adjusting the ranges from which the parameters are drawn to generate $\mathcal{F}'$ according to the measurement uncertainty, the training distribution can be modelled in such a way that the true system parameters lie inside of it with high confidence.

5.5.3 Hyperparameter Optimization

We tested MetaBO on two $2D$-hyperparameter optimization (HPO) problems for RBF-based SVMs and AdaBoost. As proposed in Wistuba et al. [Wis18], we used precomputed results of training these models on 50 datasets[1] with 144 parameter configurations (RBF kernel parameter, penalty parameter C) for the SVMs and 108 configurations (number of product terms, number of iterations) for AdaBoost. We randomly split these datasets into 35 source datasets used for training MetaBO as well as for TAF and evaluated the resulting optimization strategies on the remaining 15 datasets. To determine when to stop the meta-training of MetaBO, we performed 7-fold cross validation on the training datasets. We emphasize that MetaBO did not use more source data than TAF in this experiment, underlining again its broad applicability in situations with both scarce and abundant source data. The results (Fig. 5.5) show that MetaBO learned very data-efficient neural AFs which surpassed EI und TAF on both experiments.



(a) Simple regret (MetaBO's performance signal). **(b)** Fraction of unsolved test tasks.

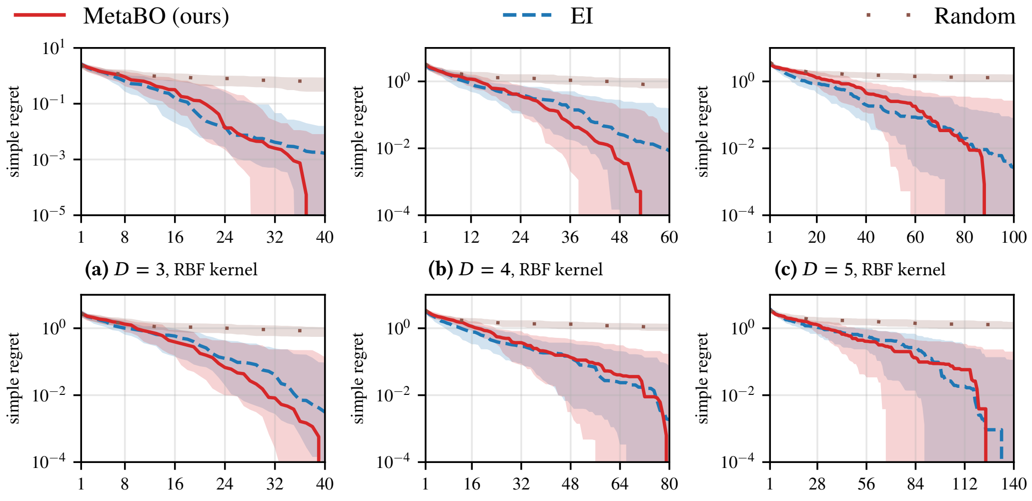
Figure 5.5: Performance on two $2D$ hyperparameter optimization tasks (SVM and AdaBoost). We trained MetaBO on precomputed data for 35 randomly chosen datasets and used the same datasets as source tasks for TAF. The remaining 15 datasets were used for this evaluation. MetaBO learned very data-efficient sampling strategies on both experiments, outperforming the benchmark methods by clear margin. Note that the optimization domain is discrete and therefore tasks can be solved exactly, corresponding to zero regret.

5.5.4 General Function Classes

Finally, we evaluated the performance of MetaBO on function classes without any particular structure except a bounded correlation lengthscale. As there is only little structure present in this function class which could be exploited in the transfer learning setting, it is desirable to obtain neural

[1] Visualizations of the objective functions can be found on http://www.hylap.org

AFs which fall back at least on the performance level of general-purpose AFs such as EI. We performed two different experiments of this type. For the first experiment, we sampled the objective functions from a GP prior with squared-exponential (RBF) kernel with lengthscales drawn uniformly from $\ell \in [0.05, 0.5]$.[1] For the second experiment, we used a GP prior with Matern-5/2 kernel with the same range of lengthscales. For the latter experiment we also used the Matern-5/2 kernel (in contrast to the RBF kernel used in all other experiments) as the kernel of the GP surrogate model to avoid model mismatch. For both types of function classes we trained MetaBO on $D = 3$ dimensional tasks and excluded the x-feature to study a dimensionality-agnostic version of MetaBO. Indeed, we evaluated the resulting neural AFs *without retraining* for dimensionalities $D \in \{3, 4, 5\}$. The results (Fig. 5.6) show that MetaBO is capable of learning neural AFs which perform better than or at least on on-par with EI on these general function classes.



(a) $D = 3$, RBF kernel (b) $D = 4$, RBF kernel (c) $D = 5$, RBF kernel

(d) $D = 3$, Matern-5/2 kernel (e) $D = 4$, Matern-5/2 kernel (f) $D = 5$, Matern-5/2 kernel

Figure 5.6: Performance of MetaBO trained on $D = 3$-dimensional objective functions sampled from a GP prior with RBF kernel (upper row) and Matern-5/2 kernel (lower row) with lengthscales drawn randomly from $\ell \in [0.05, 0.5]$. Panels (a, d) show the performance on these training distributions. As we excluded the x-feature from the neural AF inputs during training, the resulting AFs can be applied to functions of different dimensionalities. We evaluated each AF on $D = 4$ and $D = 5$ without retraining MetaBO. We report simple regret w.r.t. the best observed function value, determined separately for each function in the test set.

5.6 Conclusion and Outlook

We introduced MetaBO, a novel method for transfer learning in the framework of BO. Via a flexible meta-learning approach, we inject prior knowledge directly into the optimization strategy of BO using neural AFs. The experiments show that our method consistently outperforms existing methods, for instance in simulation-to-real settings or on hyperparameter search tasks. Our

[1] We normalized the optimization domain to $\mathcal{D} = [0, 1]^D$.

approach is broadly applicable to a wide range of practical problems, covering both the cases of scarce and abundant source data. The resulting neural AFs can represent search strategies which go far beyond the abilities of current approaches which often rely on weighted superpositions of priors over the optimization domain obtained from the source data with standard AFs. In future work, we aim to tackle the multi-task multi-fidelity setting [Val18], where we expect MetaBO's sample efficiency to be of high impact.

6 Conclusion and Outlook

In this work, we developed novel algorithms for Bayesian meta-learning (BML), with a focus on probabilistic modeling and global black-box optimization problems commonly encountered in engineering and science. In Sec. 2, we studied a generic Bayesian multitask model (Fig. 2.4) and demonstrated the central importance of accurate task posterior approximations. We then identified and addressed several shortcomings in the current practice of approximate task posterior inference in neural process (NP)-type instantiations of this model.

In Sec. 3, we showed that context aggregation in vanilla NPs ignores that task ambiguity is distributed non-uniformly in the space of context data tuples, resulting in a suboptimal parameterization of the set encoder. To alleviate this problem, we developed Bayesian context aggregation (BA), which allows learning about the task ambiguity distribution during the meta-training phase and incorporating this information into predictions in a principled manner through a weighted aggregation scheme. In a series of experiments, we demonstrated that BA leads to improved predictive performance with negligible computational overhead.

In Sec. 4, we considered further standard design choices for task posterior inference in NPs. In particular, we examined the influence of the amortized inference scheme with set encoders, the Gaussian mean-field assumption, and the variational optimization with reparameterized Euclidean gradients. We showed that these design choices can introduce a significant variational inference gap, which harms meta-training and leads to suboptimal predictive performance. By replacing this inference scheme with non-amortized, full-covariance Gaussian mixture approximations, optimized with natural gradients and trust-region step size control, we obtained significantly improved predictive performance that far exceeds the state of the art.

In addition to meta-modeling, we studied Bayesian meta-optimization, which is important for a range of applications that require the optimization of expensive black-box functions. First, we showed in Sec. 4 that our novel Bayesian meta-models provide powerful surrogate models for Bayesian optimization (BO) due to the high quality of their epistemic uncertainty estimates. In the second part of this work (Sec. 5), we developed MetaBO, a complementary approach to Bayesian meta-optimization in which the surrogate model remains untouched and inductive biases towards the meta-dataset are injected via the acquisition function.

We conclude this work with some considerations regarding the relationship between the proposed approaches, their practical applicability, as well as possible future research directions. Our results highlight the importance of accurate task posterior approximations for accurate BML. While this was to be expected from theoretical and empirical studies in Bayesian learning, such as Bayesian deep learning [Wil20, Sel23] or variational autoencoders [Kin19, Cre18], the NP model exhibits unique features, namely set-conditional inference in comparatively low-dimensional latent spaces (Sec. 2.2.5), which motivate specialized techniques such as our BA and GMM-NP. Our experimental results demonstrate the potential of such methods and motivate further research. In particular, it would be interesting to bridge the gap between amortized and non-amortized task posterior approximations as employed by BA and GMM-NP, respectively. While amortized approximations such as BA can be computed efficiently, they are typically restricted to the Gaussian mean-field case and can therefore introduce a significant amortization gap. More accurate inference techniques such as GMM-NP can reduce this gap and lead to improved predictive performance, but incur computational overhead due to their non-amortized nature. This motivates research such as the recent work by Tailor et al. [Tai23], which explores extensions of BA to obtain more expressive set-based amortization methods that go beyond the Gaussian mean-field assumption.

In the context of task posterior approximation techniques, it would also be interesting to further investigate the relative merits of different NP objectives, such as the ELBO Eq. (2.73), which spends model capacity on both predictive performance and the quality of the task posterior approximation, and NPML Eq. (2.74), which focuses exclusively on approximations that allow accurate predictions (Sec. 2.2.4.3) [Dub20]. While our results in Sec. 3 favor NPML in amortized inference settings where predictive performance is of interest, which is consistent with the literature [Gor19, Dub20], it is not yet clear whether more expressive or non-amortized task posterior approximations could still benefit from NPML, or whether a variational approach like the one used in Sec. 4 is then generally more efficient due to a tighter ELBO and a simpler optimization problem for the global parameters. Moreover, for applications that require meaningful task posterior approximations, the ELBO may be the preferable objective, since the KL term in Eq. (2.73) explicitly encourages consistency across different context sets from the same task. However, it is unclear whether such an explicit term, which is missing in NPML, is necessary, or whether the approximate task posteriors are implicitly regularized toward consistency to a sufficient degree. The same question arises for variational objectives not derived from the train-as-you-test paradigm (Sec. 2.2.4.3) [Hew18, Edw17, Le18], such as the one used for GMM-NP in Sec. 4. In this context, it would be particularly interesting to study to what extent this consistency is induced by approaches such as our BA and GMM-NP compared to simpler aggregation methods not derived from Bayesian principles (Sec. 2.2.4.2).

Another related question targeting task posterior approximations of different expressiveness and, hence, computational complexity, is whether to use the NP [Gar18c] or the CNP [Gar18b] framework (Sec. 2.2.5) for Bayesian meta-learning. Our methods proposed in Secs. 3 and 4 are rooted in the NP framework, which explicitly quantifies epistemic uncertainty about the task descriptor through approximate task posterior distributions. This allows complex predictive distributions to be specified in a natural manner through Bayesian marginalization, and naturally models correlations in the output space. On the downside, approximate task posterior inference can be computationally more demanding and less robust than the simple maximum likelihood training procedure of the CNP. In addition, unlike the CNP, the NP does not provide analytic expressions for the predictive likelihood, which can complicate downstream tasks such as BO as well as the evaluation of model performance (Sec. 2.2.3). A major drawback of the CNP, on the other hand, is that it defines a factorized predictive distribution, which prevents the computation of coherent function samples, rendering the CNP inapplicable to a range of downstream applications. While a number of methods address this issue [Bru20, Gor20, Foo20a, Mar21, Mar22, Bru23], the field of BML currently lacks a concise comparison of the predictive quality of recent NP- and CNP-based methods, which warrants a comprehensive empirical comparison.

In addition to comparing different task posterior approximation methods within the NP framework, it would be interesting to investigate the more general question of the relative merits of fully hierarchical approaches to BML and methods derived from the Bayesian multitask model Fig. 2.4. As discussed in Sec. 2.2.5, the former compute distributional estimates for all variables governing the parametric model, with examples including Bayesian instantiations of the model-agnostic meta-learning family [Fin17, Gra18, Fin18, Kim18, Rav19] and related approaches such as Amit et al. [Ami17] and Kim et al. [Kim24]. While the complexity of the resulting inference problem makes highly expressive task posterior approximation schemes such as our GMM-NP significantly more difficult to apply, it is currently unclear whether a distributional treatment of all variables, albeit with a simple task posterior approximation, is preferable [Kim24]. Furthermore, it would be interesting to study a combined approach where we use an NP-style parameterization of Fig. 2.4, but still compute a distributional estimate for the decoder weights in the spirit of fully hierarchical approaches. This would allow to retain highly expressive task posterior inference schemes for the NP task descriptors, but still enable a Bayesian treatment of the decoder parameters, which could improve the robustness of the resulting model, e.g., against out-of-distribution data [Sel23].

In the preceding paragraphs we have argued for an extensive empirical evaluation of architectural and algorithmic design choices in the field of BML. This requires accurate and standardized performance evaluation protocols in order to provide reliable comparisons, which we believe are currently lacking in the field of BML. In fact, it is well known that computing accurate estimates of performance metrics such as the approximate predictive likelihood Eq. (2.62) can be intricate and highly error-prone [Mac03, Gro15, Dub20, Far22]. This should motivate the application of more powerful and robust estimation methods such as annealed importance sampling or bidirectional Monte Carlo [Nea01, Gro15, Wu17, Cre18, Flo22], the incorporation of calibration metrics [Guo17, Kul18, Sel23], or downstream metrics [Far22] such as optimization performance as explored in Sec. 4.

An interesting aspect of meta-learning that seems to be underrepresented in the current mainstream research landscape is the application of meta-learned models to *study the physics* underlying a particular type of system in engineering or science. A simple application of such models would be to study physical processes for material constants or geometries that have not been observed in experiment [Dah23], without the need for expensive simulations or data collection. Moreover, meta-learned models could be combined with recent methods from the field of *physics-informed machine learning* [Car19, Vad21, Kar21] to discover physical concepts [Ite20, Ros20, Rai19] in the form of symbolic representations of physical laws [Udr20], symmetries [Coh16, Che19, Pol20, Kri20, Bra22], or representations of the differential equations governing their time evolution [Li21, Wan21, Ric23].

Another open question arising from our research is whether meta-learning BO should be handled by BML surrogate models, as explored in Secs. 3 and 4, or by meta-learned acquisition functions (AFs), as in MetaBO (Sec. 5). While this question can only be answered conclusively by an extensive empirical study, we believe that a combined approach has the potential to work best. Indeed, if optimization performance is the ultimate metric of interest, it should make sense to derive an approach that is optimized for exactly that metric, as this would allow learning task representations that are tailored to efficiently solve the optimization problem [Ghu23]. While MetaBO is conceptually well motivated in this sense, we believe that it would benefit from further research. An obvious shortcoming is that MetaBO handles task ambiguity in a non-Bayesian manner, as the prior of the GP surrogate model is not informed by the meta-data, and a point estimate is used for the meta-learned neural AF weights. Within the MetaBO framework, i.e., without meta-learning the surrogate model, a Bayesian treatment could be achieved by developing meta-learned AFs that maintain a Bayesian belief about the task descriptor. Alternatively, one could combine probabilistic modeling and optimization through a single shared representation, as proposed in Chen et al. [Che17] and in several works derived from our MetaBO approach [Hsi21, Mar23]. Given that recent developments in machine learning show that fewer architectural inductive biases tend to produce better results [Sut19, Vas17], this may be the most promising route.

This brings us back to the introductory discussion in Sec. 1. It remains to be seen whether the BML approaches discussed in this work will soon become obsolete by exploiting the zero- or few-shot learning capabilities of general foundation models, such as large language models (LLMs). Although the meta-learning paradigm abandons purely task-specific learning approaches in favor of achieving more general capabilities, it still employs specialized architectural and algorithmic design choices to achieve cross-task transfer. Now that the generative AI revolution is in full swing, and general transformer-based [Vas17] foundation models are becoming the go-to approach for solving all kinds of problems, it is hard to imagine that such “explicit” meta-learning approaches will survive in their current form [Sut19]. However, while it seems likely that it will soon be possible to overcome the current shortcomings of LLMs [Bow23], it is not yet clear whether general approaches will be able to continue their unprecedented growth [Bah21, Kap20, Cab23] or whether the demands on the amount [Uda24] and quality [Guo24] of training data, as well as on compute, memory, and energy [Str19, Pat21, Bal24], will become prohibitive and revive interest in specialized methods of smaller scale. Exciting times are ahead!

Bibliography

[Abd15] Abdolmaleki, Abbas; Lioutikov, Rudolf; Peters, Jan R; Lau, Nuno; Pualo Reis, Luis and Neumann, Gerhard: “Model-Based Relative Entropy Stochastic Search”. In: *Advances in Neural Information Processing Systems* (2015) (cit. on pp. 17, 62, 63).

[Aki19] Akiba, Takuya; Sano, Shotaro; Yanase, Toshihiko; Ohta, Takeru and Koyama, Masanori: “Optuna: A Next-generation Hyperparameter Optimization Framework”. In: *International Conference on Knowledge Discovery and Data Mining* (2019) (cit. on pp. 51, 132).

[Ama16] Amari, Shun-ichi: Information Geometry and Its Applications. Springer, 2016 (cit. on pp. 9, 17).

[Ama98] Amari, Shun-ichi: “Natural Gradient Works Efficiently in Learning”. In: *Neural Computation* (1998) (cit. on pp. 17, 62, 63).

[Ami17] Amit, Ron and Meir, Ron: “Meta-Learning by Adjusting Priors Based on Extended PAC-Bayes Theory”. In: *International Conference on Machine Learning* (2017) (cit. on pp. 40, 93).

[Amo23] Amos, Brandon: “Tutorial on Amortized Optimization”. In: *Foundations and Trends in Machine Learning* (2023) (cit. on p. 17).

[And16] Andrychowicz, Marcin; Denil, Misha; Colmenarejo, Sergio Gomez; Hoffman, Matthew W.; Pfau, David; Schaul, Tom and Freitas, Nando de: “Learning to Learn by Gradient Descent by Gradient Descent”. In: *Advances in Neural Information Processing Systems* (2016) (cit. on pp. 38, 44, 77).

[Ant19] Antoniou, Antreas; Edwards, Harrison and Storkey, Amos: “How to train your MAML”. In: *International Conference on Learning Representations* (2019) (cit. on p. 39).

[Are18] Arenz, Oleg; Neumann, Gerhard and Zhong, Mingjun: “Efficient Gradient-Free Variational Inference using Policy Search”. In: *International Conference on Machine Learning* (2018) (cit. on pp. 62, 63).

[Are20] Arenz, O.; Zhong, M. and G., Neumann: “Trust-Region Variational Inference with Gaussian Mixture Models”. In: *Journal of Machine Learning Research* (2020) (cit. on p. 17).

[Are23] Arenz, Oleg; Dahlinger, Philipp; Ye, Zihan; Volpp, Michael and Neumann, Gerhard: “A Unified Perspective on Natural Gradient Variational Inference with Gaussian Mixture Models”. In: *Transactions on Machine Learning Research* (2023) (cit. on pp. 3, 19, 60–63, 65, 66, 145, 146, 148, 150, 155, 156, 163).

[Att00] Attias, Hagai: “A variational Bayesian framework for graphical models”. In: *Advances in Neural Information Processing Systems* (2000) (cit. on p. 62).

[Aze24] Azerbayev, Zhangir; Schoelkopf, Hailey; Paster, Keiran; Santos, Marco Dos; McAleer, Stephen Marcus; Jiang, Albert Q.; Deng, Jia; Biderman, Stella and Welleck, Sean: “Llemma: An Open Language Model for Mathematics”. In: *International Conference on Learning Representations* (2024) (cit. on p. 1).

[Bah21] Bahri, Yasaman; Dyer, Ethan; Kaplan, Jared; Lee, Jaehoon and Sharma, Utkarsh: “Explaining Neural Scaling Laws”. In: *arXiv* (2021) (cit. on p. 94).

[Bak03] Bakker, Bart and Heskes, Tom: “Task Clustering and Gating for Bayesian Multitask Learning”. In: *Journal of Machine Learning Research* (2003) (cit. on pp. 4, 26, 28, 44, 61).

[Bal20] Balandat, Maximilian; Karrer, Brian; Jiang, Daniel R.; Daulton, Samuel; Letham, Benjamin; Wilson, Andrew Gordon and Bakshy, Eytan: “BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimization”. In: *Advances in Neural Information Processing Systems* (2020) (cit. on p. 22).

[Bal24] Balne, Charith Chandra Sai; Bhaduri, Sreyoshi; Roy, Tamoghna; Jain, Vinija and Chadha, Aman: “Parameter Efficient Fine Tuning: A Comprehensive Analysis Across Applications”. In: *arXiv* (2024) (cit. on pp. 2, 94).

[Ban17] Bansal, Somil; Calandra, Roberto; Xiao, Ted; Levine, Sergey and Tomlin, Claire J.: “Goal-driven Dynamics Learning via Bayesian Optimization”. In: *IEEE Annual Conference on Decision and Control* (2017) (cit. on p. 85).

[Bar13] Bardenet, Rémi; Brendel, Mátyás; Kégl, Balázs and Sebag, Michèle: “Collaborative Hyperparameter Tuning”. In: *International Conference on Machine Learning* (2013) (cit. on pp. 77, 128).

[Bay63] Bayes, Thomas: “An essay towards solving a problem in the doctrine of chances. By the late Rev. Mr. Bayes, FRS communicated by Mr. Price, in a letter to John Canton, AMFR S”. In: *Philosophical Transactions of the Royal Society of London* (1763) (cit. on p. 8).

[Bec19] Becker, Philipp; Pandya, Harit; Gebhardt, Gregor H. W.; Zhao, Cheng; Taylor, C. James and Neumann, Gerhard: “Recurrent Kalman Networks: Factorized Inference in High-Dimensional Deep Feature Spaces”. In: *International Conference on Machine Learning* (2019) (cit. on pp. 44, 50).

[Ben13] Bengio, Yoshua; Courville, Aaron and Vincent, Pascal: “Representation Learning: A Review and New Perspectives”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2013) (cit. on pp. 1, 23).

[Ben91] Bengio, Yoshua; Bengio, Samy and Cloutier, Jocelyn: “Learning a Synaptic Learning Rule”. In: *International Joint Conference on Neural Networks* (1991) (cit. on pp. 38, 44, 61, 77).

[Ber91] Berger, J. and Bernardo, Jose: "On the development of reference priors". In: *Bayesian Statistics* (1991) (cit. on p. 13).

[Bet15] Betancourt, Michael: "The Fundamental Incompatibility of Scalable Hamiltonian Monte Carlo and Naive Data Subsampling". In: *International Conference on Machine Learning* (2015) (cit. on p. 16).

[Bet18] Betancourt, Michael: "A Conceptual Introduction to Hamiltonian Monte Carlo". In: *arXiv* (2018) (cit. on pp. 14–16, 18).

[Bie20] Biewald, Lukas: Experiment Tracking with Weights and Biases. 2020. url: https://www.wandb.com/ (cit. on p. 149).

[Bis06] Bishop, Christopher M.: Pattern Recognition and Machine Learning. Springer, 2006 (cit. on pp. 3, 5–7, 13–16, 18, 19, 28, 42, 51, 126, 146, 147).

[Bis23] Bishop, Christopher Michael and Bishop, Hugh: Deep Learning - Foundations and Concepts. Springer, 2023 (cit. on pp. 1, 13, 23).

[Bis94] Bishop, Christopher M.: Mixture density networks. Technical Report. 1994 (cit. on pp. 11, 18).

[Bis97] Bishop, Christopher; Lawrence, Neil; Jaakkola, Tommi and Jordan, Michael: "Approximating Posterior Distributions in Belief Networks Using Mixtures". In: *Advances in Neural Information Processing Systems* (1997) (cit. on p. 18).

[Ble17] Blei, David M.; Kucukelbir, Alp and McAuliffe, Jon D.: "Variational Inference: A Review for Statisticians". In: *Journal of the American Statistical Association* (2017) (cit. on p. 18).

[Blu15] Blundell, Charles; Cornebise, Julien; Kavukcuoglu, Koray and Wierstra, Daan: "Weight Uncertainty in Neural Networks". In: *International Conference on Machine Learning* (2015) (cit. on pp. 18, 62).

[Bod24] Bodnar, Cristian et al.: "Aurora: A Foundation Model of the Atmosphere". In: *arXiv* (2024) (cit. on p. 1).

[Bom21] Bommasani, Rishi et al.: "On the Opportunities and Risks of Foundation Models". In: *arXiv* (2021) (cit. on p. 1).

[Bow23] Bowman, Samuel R.: "Eight Things to Know about Large Language Models". In: *arXiv* (2023) (cit. on pp. 2, 94).

[Bra22] Brandstetter, Johannes; Welling, Max and Worrall, Daniel E: "Lie Point Symmetry Data Augmentation for Neural PDE Solvers". In: *International Conference on Machine Learning* (2022) (cit. on p. 94).

[Bro10] Brochu, Eric; Cora, Vlad M. and Freitas, Nando de: "A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning". In: *arXiv* (2010) (cit. on p. 76).

[Bro20] Brown, Tom et al.: "Language Models are Few-Shot Learners". In: *Advances in Neural Information Processing Systems* (2020) (cit. on p. 1).

[Bro21] Bronstein, Michael M.; Bruna, Joan; Cohen, Taco and Velickovic, Petar: "Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges". In: *arXiv* (2021) (cit. on p. 1).

[Bru20] Bruinsma, Wessel P.; Requeima, James; Foong, Andrew Y. K.; Gordon, Jonathan and Turner, Richard E.: "The Gaussian Neural Process". In: *Symposium on Advances in Approximate Bayesian Inference* (2020) (cit. on pp. 38, 93).

[Bru23] Bruinsma, Wessel; Markou, Stratis; Requeima, James; Foong, Andrew Y. K.; Andersson, Tom; Vaughan, Anna; Buonomo, Anthony; Hosking, Scott and Turner, Richard E: "Autoregressive Conditional Neural Processes". In: *International Conference on Learning Representations* (2023) (cit. on pp. 38, 93).

[Bub23] Bubeck, Sébastien et al.: "Sparks of Artificial General Intelligence: Early experiments with GPT-4". In: *arXiv* (2023) (cit. on p. 1).

[Bur16] Burda, Yuri; Grosse, Roger B. and Salakhutdinov, Ruslan: "Importance Weighted Autoencoders". In: *International Conference on Learning Representations* (2016) (cit. on p. 33).

[Cab23] Caballero, Ethan; Gupta, Kshitij; Rish, Irina and Krueger, David: "Broken Neural Scaling Laws". In: *International Conference on Learning Representations* (2023) (cit. on pp. 1, 94).

[Cal16] Calandra, R.; Peters, J.; Rasmussen, C. E. and Deisenroth, M. P.: "Manifold Gaussian Processes for Regression". In: *International Joint Conference on Neural Networks* (2016) (cit. on p. 43).

[Cam53] Cam, Lucien le: "On some asymptotic properties of maximum likelihood estimates and related Bayes' estimates". In: *University of California Publications in Statistics* (1953) (cit. on p. 9).

[Car19] Carleo, Giuseppe; Cirac, Ignacio; Cranmer, Kyle; Daudet, Laurent; Schuld, Maria; Tishby, Naftali; Vogt-Maranto, Leslie and Zdeborová, Lenka: "Machine learning and the physical sciences". In: *Reviews of Modern Physics* (2019) (cit. on p. 93).

[Caz11] Cazzolato, Benjamin Seth and Prime, Zebb: "On the Dynamics of the Furuta Pendulum". In: *Journal of Control Science and Engineering* (2011) (cit. on pp. 132, 155).

[Che17] Chen, Yutian; Hoffman, Matthew W.; Colmenarejo, Sergio Gómez; Denil, Misha; Lillicrap, Timothy P.; Botvinick, Matt and Freitas, Nando de: "Learning to Learn without Gradient Descent by Gradient Descent". In: *International Conference on Machine Learning* (2017) (cit. on pp. 38, 44, 77, 94).

[Che19] Cheng, Miranda C. N.; Anagiannis, Vassilis; Weiler, Maurice; Haan, Pim de; Cohen, Taco S. and Welling, Max: "Covariance in Physics and Convolutional Neural Networks". In: *ICML Workshop on Theoretical Physics for Deep Learning* (2019) (cit. on p. 94).

[Che21] Chen, Mark et al.: “Evaluating Large Language Models Trained on Code”. In: *arXiv* (2021) (cit. on p. 1).

[Chu18] Chua, Kurtland; Calandra, Roberto; McAllister, Rowan and Levine, Sergey: “Deep reinforcement learning in a handful of trials using probabilistic dynamics models”. In: *Advances in Neural Information Processing Systems* (2018) (cit. on p. 60).

[Coh16] Cohen, Taco and Welling, Max: “Group Equivariant Convolutional Networks”. In: *International Conference on Machine Learning* (2016) (cit. on p. 93).

[Cox46] Cox, R. T.: “Probability, Frequency and Reasonable Expectation”. In: *American Journal of Physics* (1946) (cit. on p. 7).

[Cox61] Cox, Richard T: Algebra of probable inference. Johns Hopkins University Press, 1961 (cit. on pp. 7, 8).

[Cre18] Cremer, Chris; Li, Xuechen and Duvenaud, David: “Inference Suboptimality in Variational Autoencoders”. In: *International Conference on Machine Learning* (2018) (cit. on pp. 18, 61, 65, 92, 93).

[Cur44] Curry, Haskell B.: “The method of steepest descent for non-linear minimization problems”. In: *Quarterly of Applied Mathematics* (1944) (cit. on p. 13).

[Dah23] Dahlinger, Philipp; Freymuth, Niklas; Hoang, Tai; Volpp, Michael and Neumann, Gerhard: “Latent Task-Specific Graph Network Simulators”. In: *arXiv* (2023) (cit. on pp. 3, 93).

[Dam13] Damianou, Andreas and Lawrence, Neil: “Deep Gaussian Processes”. In: *International Conference on Artificial Intelligence and Statistics* (2013) (cit. on pp. 43, 128).

[Dan16] Daniel, Christian; Taylor, Jonathan and Nowozin, Sebastian: “Learning Step Size Controllers for Robust Neural Network Training”. In: *AAAI Conference on Artificial Intelligence* (2016) (cit. on p. 77).

[Dei11a] Deisenroth, Marc Peter; Neumann, Gerhard and Peters, Jan: “A Survey on Policy Search for Robotics”. In: *Foundations and Trends in Robotics* (2011) (cit. on pp. 60, 72).

[Dei11b] Deisenroth, MP. and Rasmussen, CE.: “PILCO: A Model-Based and Data-Efficient Approach to Policy Search”. In: *International Conference on Machine Learning* (2011) (cit. on p. 50).

[Dem77] Dempster, A. P.; Laird, N. M. and Rubin, D. B.: “Maximum likelihood from incomplete data via the EM algorithm”. In: *Journal of the Royal Statistical Society* (1977) (cit. on p. 19).

[Den09] Deng, Jia; Dong, Wei; Socher, Richard; Li, Li-Jia; Li, Kai and Fei-Fei, Li: “ImageNet: A large-scale hierarchical image database”. In: *IEEE Conference on Computer Vision and Pattern Recognition* (2009) (cit. on p. 1).

[Dev19] Devlin, Jacob; Chang, Ming-Wei; Lee, Kenton and Toutanova, Kristina: “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (2019) (cit. on p. 1).

[Dev96] Devroye, Luc: “Random Variate Generation in One Line of Code”. In: *Conference on Winter Simulation* (1996) (cit. on p. 17).

[Din17] Dinh, Laurent; Sohl-Dickstein, Jascha and Bengio, Samy: “Density Estimation using Real NVP”. In: *International Conference on Learning Representations* (2017) (cit. on p. 1).

[Dir30] Dirac, P. A. M.: The principles of quantum mechanics. Oxford University Press, 1930 (cit. on p. 10).

[Doe19] Doerr, Andreas; Volpp, Michael; Toussaint, Marc; Sebastian, Trimpe and Daniel, Christian: “Trajectory-Based Off-Policy Deep Reinforcement Learning”. In: *International Conference on Machine Learning* (2019) (cit. on p. 4).

[Dom18] Domke, Justin and Sheldon, Daniel: “Importance Weighting and Variational Inference”. In: *arXiv* (2018) (cit. on p. 36).

[Doo49] Doob, J. L.: “Application of the theory of martingales”. In: *Actes du Colloque International Le Calcul des Probabilités et ses applications* (1949) (cit. on pp. 9, 10).

[Dos21] Dosovitskiy, Alexey et al.: “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”. In: *International Conference on Learning Representations* (2021) (cit. on p. 1).

[Dra18] Draxler, Felix; Veschgini, Kambis; Salmhofer, Manfred and Hamprecht, Fred: “Essentially No Barriers in Neural Network Energy Landscape”. In: *International Conference on Machine Learning* (2018) (cit. on p. 14).

[Dua87] Duane, Simon; Kennedy, A. D.; Pendleton, Brian J. and Roweth, Duncan: “Hybrid Monte Carlo”. In: *Physics Letters B* (1987) (cit. on p. 16).

[Dub20] Dubois, Yann; Gordon, Jonathan and Foong, Andrew YK: Neural Process Family. 2020. url: http://yanndubs.github.io/Neural-Process-Family/ (cit. on pp. 33, 36, 92, 93).

[Duc11] Duchi, John; Hazan, Elad and Singer, Yoram: “Adaptive Subgradient Methods for Online Learning and Stochastic Optimization”. In: *Journal of Machine Learning Research* (2011) (cit. on pp. 13, 17, 37).

[Edw17] Edwards, Harrison A. and Storkey, Amos J.: “Towards a Neural Statistician”. In: *International Conference on Learning Representations* (2017) (cit. on pp. 38, 44, 61, 92).

[Fan16] Fan, Zhou: Lecture Notes on Statistical Inference: MLE under model misspecification. 2016. url: https://web.stanford.edu/class/archive/stats/stats200/stats200.1172/Lecture16.pdf (cit. on p. 9).

[Far20] Farquhar, Sebastian; Smith, Lewis and Gal, Yarin: “Liberty or Depth: Deep Bayesian Neural Nets Do Not Need Complex Weight Posterior Approximations”. In: *Conference on Neural Information Processing Systems* (2020) (cit. on p. 18).

[Far22] Farquhar, Sebastian: “Understanding Approximation for Bayesian Inference in Neural Networks”. PhD thesis. Oxford University, 2022 (cit. on pp. 33, 93).

[Fei06] Fei-Fei, Li; Fergus, R. and Perona, P.: “One-shot Learning of Object Categories”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2006) (cit. on p. 44).

[Feu15a] Feurer, Matthias; Klein, Aaron; Eggensperger, Katharina; Springenberg, Jost Tobias; Blum, Manuel and Hutter, Frank: “Efficient and Robust Automated Machine Learning”. In: *Advances in Neural Information Processing Systems* (2015) (cit. on p. 78).

[Feu15b] Feurer, Matthias; Springenberg, Jost Tobias and Hutter, Frank: “Initializing Bayesian Hyperparameter Optimization via Meta-Learning”. In: *AAAI Conference on Artificial Intelligence* (2015) (cit. on p. 78).

[Feu18] Feurer, Matthias; Letham, Benjamin and Bakshy, Eytan: “Scalable Meta-Learning for Bayesian Optimization”. In: *arXiv* (2018) (cit. on pp. 76, 78).

[Fin17] Finn, Chelsea; Abbeel, Pieter and Levine, Sergey: “Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks”. In: *International Conference on Machine Learning* (2017) (cit. on pp. 38, 44, 61, 68, 77, 93).

[Fin18] Finn, Chelsea; Xu, Kelvin and Levine, Sergey: “Probabilistic Model-Agnostic Meta-Learning”. In: *Advances in Neural Information Processing Systems* (2018) (cit. on pp. 40, 44, 61, 68, 93).

[Fin19] Finn, Chelsea; Rajeswaran, Aravind; Kakade, Sham and Levine, Sergey: “Online Meta-Learning”. In: *International Conference on Machine Learning* (2019) (cit. on p. 39).

[Fin30] Finetti, Bruno de: “Funzione caratteristica di un fenomeno aleatorio”. In: *Memorie della R. Accademia dei Lincei* (1930) (cit. on p. 6).

[Fin37] Finetti, Bruno de: “La prévision : ses lois logiques, ses sources subjectives”. In: *Annales de l’institut Henri Poincaré* (1937) (cit. on p. 6).

[Fle19] Flennerhag, Sebastian; Moreno, Pablo Garcia; Lawrence, Neil and Damianou, Andreas: “Transferring Knowledge across Learning Processes”. In: *International Conference on Learning Representations* (2019) (cit. on p. 77).

[Flo22] Florin, Lucas: “Annealed Importance Sampling for Neural Process Training”. Master thesis. Karlsruhe Institute of Technology, Karlsruhe, Germany, 2022 (cit. on pp. 4, 93).

[Foo20a] Foong, Andrew; Bruinsma, Wessel; Gordon, Jonathan; Dubois, Yann; Requeima, James and Turner, Richard: "Meta-Learning Stationary Stochastic Process Prediction with Convolutional Neural Processes". In: *Advances in Neural Information Processing Systems* (2020) (cit. on pp. 8, 35, 36, 38, 93).

[Foo20b] Foong, Andrew; Burt, David; Li, Yingzhen and Turner, Richard: "On the Expressiveness of Approximate Inference in Bayesian Neural Networks". In: *Advances in Neural Information Processing Systems* (2020) (cit. on p. 18).

[For08] Forrester, Alexander I. J.; Sobester, Andras and Keane, Andy J.: Engineering Design via Surrogate Modelling - A Practical Guide. Wiley, 2008 (cit. on pp. 71, 154).

[For22a] Fortuin, Vincent: "Priors in Bayesian Deep Learning: A Review". In: *International Statistical Review* (2022) (cit. on pp. 13, 14).

[For22b] Fortuin, Vincent; Garriga-Alonso, Adrià; Ober, Sebastian W.; Wenzel, Florian; Ratsch, Gunnar; Turner, Richard E; Wilk, Mark van der and Aitchison, Laurence: "Bayesian Neural Network Priors Revisited". In: *International Conference on Learning Representations* (2022) (cit. on p. 13).

[Frö20] Fröhlich, Lukas P.; Klenske, Edgar D.; Daniel, Christian G. and Zeilinger, Melanie N.: "Bayesian Optimization for Policy Search in High-Dimensional Systems via Automatic Domain Selection". In: *arXiv* (2020) (cit. on p. 85).

[Fur92] Furuta, K.; Yamakita, M. and Kobayashi, S.: "Swing-up Control of Inverted Pendulum Using Pseudo-State Feedback". In: *Journal of Systems and Control Engineering* (1992) (cit. on pp. 54, 72, 85, 132).

[Gal14] Gallet, Ana; Volpp, Michael and Lengerer, Wolfgang: Standard development process for physical models used in real time applications based on the example of an exhaust pipe model. Technical Report. 2014 (cit. on p. 1).

[Gal16] Gal, Yarin and Ghahramani, Zoubin: "Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning". In: *International Conference on Machine Learning* (2016) (cit. on pp. 43, 128).

[Gao22] Gao, Ning; Ziesche, Hanna; Vien, Ngo Anh; Volpp, Michael and Neumann, Gerhard: "What Matters for Meta-Learning Vision Regression Tasks?" In: *IEEE Conference on Computer Vision and Pattern Recognition* (2022) (cit. on p. 3).

[Gar18a] Garipov, Timur; Izmailov, Pavel; Podoprikhin, Dmitrii; Vetrov, Dmitry P and Wilson, Andrew G: "Loss Surfaces, Mode Connectivity, and Fast Ensembling of DNNs". In: *Advances in Neural Information Processing Systems* (2018) (cit. on pp. 11, 14, 18).

[Gar18b] Garnelo, Marta; Rosenbaum, Dan; Maddison, Christopher; Ramalho, Tiago; Saxton, David; Shanahan, Murray; Teh, Yee Whye; Rezende, Danilo and Eslami, S. M. Ali: "Conditional Neural Processes". In: *International Conference on Machine Learning* (2018) (cit. on pp. 35, 37, 38, 43, 44, 47, 49, 51, 61, 92, 128, 129).

[Gar18c] Garnelo, Marta; Schwarz, Jonathan; Rosenbaum, Dan; Viola, Fabio; Rezende, Danilo Jimenez; Eslami, S. M. Ali and Teh, Yee Whye: “Neural Processes”. In: *ICML Workshop on Theoretical Foundations and Applications of Deep Generative Models* (2018) (cit. on pp. 3–5, 8, 33–37, 41, 42, 44–46, 49, 51, 59–62, 64, 68, 71, 72, 78, 92, 128, 129, 147–149, 151).

[Gar23] Garg, Shivam; Tsipras, Dimitris; Liang, Percy and Valiant, Gregory: “What Can Transformers Learn In-Context? A Case Study of Simple Function Classes”. In: *arXiv* (2023) (cit. on p. 1).

[Gel13] Gelman, A.; Carlin, J.B.; Stern, H.S.; Dunson, D.B.; Vehtari, A. and Rubin, D.B.: Bayesian Data Analysis. Taylor & Francis, 2013 (cit. on pp. 3, 5–7, 9, 14, 15, 28, 40, 62).

[Gem84] Geman, Stuart and Geman, Donald: “Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (1984) (cit. on p. 16).

[Ger12] Gershman, Samuel J.; Hoffman, Matthew D. and Blei, David M.: “Nonparametric Variational Inference”. In: *International Conference on Machine Learning* (2012) (cit. on p. 18).

[Ger14] Gershman, Samuel J. and Goodman, Noah D.: “Amortized Inference in Probabilistic Reasoning”. In: *Cognitive Science* (2014) (cit. on p. 17).

[Gey13] Geyer, Charles J.: Lecture Notes on Statistics: The Sandwich Estimator. 2013. url: https://www.stat.umn.edu/geyer/5601/notes/sand.pdf (cit. on p. 9).

[Gho20] Ghosh, Partha; Sajjadi, Mehdi S. M.; Vergari, Antonio; Black, Michael and Schölkopf, Bernhard: “From Variational to Deterministic Autoencoders”. In: *International Conference on Learning Representations* (2020) (cit. on p. 38).

[Ghu23] Ghugare, Raj; Bharadhwaj, Homanga; Eysenbach, Benjamin; Levine, Sergey and Salakhutdinov, Russ: “Simplifying Model-based RL: Learning Representations, Latent-space Models, and Policies with One Objective”. In: *International Conference on Learning Representations* (2023) (cit. on p. 94).

[Gok19] Gokaslan, Aaron and Cohen, Vanya: OpenWebText Corpus. 2019. url: http://Skylion007.github.io/OpenWebTextCorpus (cit. on p. 1).

[Gol17] Golovin, Daniel; Solnik, Benjamin; Moitra, Subhodeep; Kochanski, Greg; Karro, John and Sculley, D.: “Google Vizier: A Service for Black-Box Optimization”. In: *International Conference on Knowledge Discovery and Data Mining* (2017) (cit. on pp. 61, 78, 128).

[Goo14] Goodfellow, Ian; Pouget-Abadie, Jean; Mirza, Mehdi; Xu, Bing; Warde-Farley, David; Ozair, Sherjil; Courville, Aaron and Bengio, Yoshua: “Generative Adversarial Nets”. In: *Advances in Neural Information Processing Systems* (2014) (cit. on p. 1).

[Goo16] Goodfellow, Ian; Bengio, Yoshua and Courville, Aaron: Deep Learning. MIT Press, 2016 (cit. on pp. 1, 13, 20).

[Gor19] Gordon, Jonathan; Bronskill, John; Bauer, Matthias; Nowozin, Sebastian and Turner, Richard E.: "Meta-Learning Probabilistic Inference for Prediction". In: *International Conference on Learning Representations* (2019) (cit. on pp. 36, 38, 42, 44–47, 49, 61, 92).

[Gor20] Gordon, Jonathan; Bruinsma, Wessel P.; Foong, Andrew Y. K.; Requeima, James; Dubois, Yann and Turner, Richard E.: "Convolutional Conditional Neural Processes". In: *International Conference on Learning Representations* (2020) (cit. on pp. 35, 38, 41, 44, 52, 93, 130).

[GPy12] GPy: GPy: A Gaussian process framework in python. 2012. url: http://github.com/SheffieldML/GPy (cit. on p. 174).

[Gra11] Graves, Alex: "Practical Variational Inference for Neural Networks". In: *Advances in Neural Information Processing Systems* (2011) (cit. on p. 18).

[Gra18] Grant, Erin; Finn, Chelsea; Levine, Sergey; Darrell, Trevor and Griffiths, Thomas L.: "Recasting Gradient-Based Meta-Learning as Hierarchical Bayes". In: *International Conference on Learning Representations* (2018) (cit. on pp. 40, 44, 61, 93).

[Gro15] Grosse, Roger B.; Ghahramani, Zoubin and Adams, Ryan P.: "Sandwiching the marginal likelihood using bidirectional Monte Carlo". In: *arXiv* (2015) (cit. on pp. 15, 32, 33, 93).

[Guo17] Guo, Chuan; Pleiss, Geoff; Sun, Yu and Weinberger, Kilian Q.: "On Calibration of Modern Neural Networks". In: *International Conference on Machine Learning* (2017) (cit. on p. 93).

[Guo24] Guo, Yanzhu; Shang, Guokan; Vazirgiannis, Michalis and Clavel, Chloé: "The Curious Decline of Linguistic Diversity: Training Language Models on Synthetic Text". In: *arXiv* (2024) (cit. on p. 94).

[Ha17] Ha, David; Dai, Andrew M. and Le, Quoc V.: "HyperNetworks". In: *International Conference on Learning Representations* (2017) (cit. on p. 38).

[Has70] Hastings, W. K.: "Monte Carlo sampling methods using Markov chains and their applications". In: *Biometrika* (1970) (cit. on p. 16).

[He16] He, Kaiming; Zhang, Xiangyu; Ren, Shaoqing and Sun, Jian: "Deep Residual Learning for Image Recognition". In: *IEEE Conference on Computer Vision and Pattern Recognition* (2016) (cit. on p. 1).

[Hen12] Hennig, Philipp and Schuler, Christian J.: "Entropy Search for Information-efficient Global Optimization". In: *Journal of Machine Learning Research* (2012) (cit. on p. 21).

[Hen14] Henrández-Lobato, José Miguel; Hoffman, Matthew W. and Ghahramani, Zoubin: “Predictive Entropy Search for Efficient Global Optimization of Black-box Functions”. In: *Advances in Neural Information Processing Systems* (2014) (cit. on p. 21).

[Hes00] Heskes, Tom: “Empirical Bayes for Learning to Learn”. In: *International Conference on Machine Learning* (2000) (cit. on pp. 4, 26, 44, 45, 61).

[Hew18] Hewitt, Luke B.; Nye, Maxwell I.; Gane, Andreea; Jaakkola, Tommi S. and Tenenbaum, Joshua B.: “The Variational Homoencoder: learning to Infer High Capacity Generative Models from Few Examples”. In: *Conference on Uncertainty in Artificial Intelligence* (2018) (cit. on pp. 38, 61, 92).

[Hew55] Hewitt, Edwin Shields and Savage, Leonard J.: “Symmetric measures on Cartesian products”. In: *Transactions of the American Mathematical Society* (1955) (cit. on p. 6).

[Hin08] Hinton, Geoffrey E. and Salakhutdinov, Russ R.: “Using Deep Belief Nets to Learn Covariance Kernels for Gaussian Processes”. In: *Advances in Neural Information Processing Systems* (2008) (cit. on p. 43).

[Hin12] Hinton, Geoffrey: Lectures on Neural Networks for Machine Learning, Lecture 6. 2012. url: https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture%5C_slides%5C_lec6.pdf (cit. on p. 13).

[Hin93] Hinton, Geoffrey E. and Camp, Drew van: “Keeping the Neural Networks Simple by Minimizing the Description Length of the Weights”. In: *Conference on Computational Learning Theory* (1993) (cit. on pp. 18, 43).

[Ho20] Ho, Jonathan; Jain, Ajay and Abbeel, Pieter: “Denoising Diffusion Probabilistic Models”. In: *arXiv* (2020) (cit. on p. 1).

[Hoc01] Hochreiter, Sepp; Younger, A. Steven and Conwell, Peter R.: “Learning to Learn Using Gradient Descent”. In: *International Conference on Artificial Neural Networks* (2001) (cit. on pp. 38, 44, 77).

[Hof13] Hoffman, Matthew D.; Blei, David M.; Wang, Chong and Paisley, John W.: “Stochastic variational inference”. In: *Journal of Machine Learning Research* (2013) (cit. on pp. 17, 18, 62).

[Hon08] Honkela, Antti; Tornio, Matti; Raiko, Tapani and Karhunen, Juha: “Natural Conjugate Gradient in Variational Inference”. In: *Neural Information Processing* (2008) (cit. on pp. 17, 18).

[Hor89] Hornik, Kurt; Stinchcombe, Maxwell and White, Halbert: “Multilayer feedforward networks are universal approximators”. In: *Neural Networks* (1989) (cit. on p. 13).

[Hos22] Hospedales, T.; Antoniou, A.; Micaelli, P. and Storkey, A.: “Meta-Learning in Neural Networks: a Survey”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2022) (cit. on pp. 1–3, 23, 37, 60, 61).

[Hsi21] Hsieh, Bing-Jing; Hsieh, Ping-Chun and Liu, Xi: “Reinforced Few-Shot Acquisition Function Learning for Bayesian Optimization”. In: *Advances in Neural Information Processing Systems* (2021) (cit. on p. 94).

[Hu21] Hu, Edward J.; Shen, Yelong; Wallis, Phillip; Allen-Zhu, Zeyuan; Li, Yuanzhi; Wang, Shean; Wang, Lu and Chen, Weizhu: “LoRA: Low-Rank Adaptation of Large Language Models”. In: *arXiv* (2021) (cit. on p. 1).

[Hui21] Huisman, Mike; Rijn, Jan N. van and Plaat, Aske: “A survey of deep meta-learning”. In: *Artificial Intelligence Review* (2021) (cit. on pp. 1–3, 23, 37, 38).

[Hül21] Hüllermeier, Eyke and Waegeman, Willem: “Aleatoric and epistemic uncertainty in machine learning: an introduction to concepts and methods”. In: *Machine Learning* (2021) (cit. on pp. 6, 11).

[Hut11] Hutter, Frank; Hoos, Holger H. and Leyton-Brown, Kevin: “Sequential Model-Based Optimization for General Algorithm Configuration”. In: *Learning and Intelligent Optimization* (2011) (cit. on p. 21).

[Ite20] Iten, Raban; Metger, Tony; Wilming, Henrik; Rio, Lídia del and Renner, Renato: “Discovering Physical Concepts with Neural Networks”. In: *Physics Review Letters* (2020) (cit. on p. 93).

[Izm21] Izmailov, Pavel; Vikram, Sharad; Hoffman, Matthew D and Wilson, Andrew Gordon Gordon: “What Are Bayesian Neural Network Posteriors Really Like?” In: *International Conference on Machine Learning* (2021) (cit. on pp. 14, 16).

[Jaa98] Jaakkola, Tommi S. and Jordan, Michael I.: “Improving the Mean Field Approximation Via the Use of Mixture Distributions”. In: *Learning in Graphical Models* (1998) (cit. on p. 18).

[Jay68] Jaynes, Edwin T.: “Prior Probabilities”. In: *IEEE Transactions on Systems Science and Cybernetics* (1968) (cit. on p. 13).

[Jef46] Jeffreys, Harold: “An invariant form for the prior probability in estimation problems”. In: *Proceedings of the Royal Society of London* (1946) (cit. on p. 13).

[Jen06] Jensen, J. L. W. V.: “Sur les fonctions convexes et les inégalités entre les valeurs moyennes”. In: *Acta Mathematica* (1906) (cit. on p. 33).

[Jha23] Jha, Saurav; Gong, Dong; Wang, Xuesong; Turner, Richard E. and Yao, Lina: The Neural Process Family: Survey, Applications and Perspectives. 2023 (cit. on p. 38).

[Jon98] Jones, Donald R.; Schonlau, Matthias and Welch, William J.: “Efficient Global Optimization of Expensive Black-Box Functions”. In: *Journal of Global Optimization* (1998) (cit. on p. 21).

[Jor99] Jordan, Michael I.; Ghahramani, Zoubin; Jaakkola, Tommi S. and Saul, Lawrence K.: “An Introduction to Variational Methods for Graphical Models”. In: *Machine Learning* (1999) (cit. on p. 16).

[Jum21] Jumper, John M. et al.: “Highly accurate protein structure prediction with AlphaFold”. In: *Nature* (2021) (cit. on p. 1).

[Kan18] Kandasamy, Kirthevasan; Krishnamurthy, Akshay; Schneider, Jeff and Poczos, Barnabas: “Parallelised Bayesian Optimisation via Thompson Sampling”. In: *International Conference on Artificial Intelligence and Statistics* (2018) (cit. on p. 21).

[Kap20] Kaplan, Jared; McCandlish, Sam; Henighan, Tom; Brown, Tom B.; Chess, Benjamin; Child, Rewon; Gray, Scott; Radford, Alec; Wu, Jeffrey and Amodei, Dario: “Scaling Laws for Neural Language Models”. In: *arXiv* (2020) (cit. on pp. 1, 94).

[Kar21] Karniadakis, George Em; Kevrekidis, Ioannis G.; Lu, Lu; Perdikaris, Paris; Wang, Sifan and Yang, Liu: “Physics-informed machine learning”. In: *Nature Reviews Physics* (2021) (cit. on p. 93).

[Kas96] Kass, Robert E. and Wasserman, Larry: “The Selection of Prior Distributions by Formal Rules”. In: *Journal of the American Statistical Association* (1996) (cit. on p. 13).

[Kha17] Khan, Mohammad and Lin, Wu: “Conjugate-Computation Variational Inference : Converting Variational Inference in Non-Conjugate Models to Inferences in Conjugate Models”. In: *International Conference on Artificial Intelligence and Statistics* (2017) (cit. on p. 17).

[Kha18a] Khan, Mohammad Emtiyaz and Nielsen, Didrik: “Fast yet Simple Natural-Gradient Descent for Variational Inference in Complex Models”. In: *International Symposium on Information Theory and its Applications* (2018) (cit. on pp. 61, 62, 65).

[Kha18b] Khan, Mohammad Emtiyaz; Nielsen, Didrik; Tangkaratt, Voot; Lin, Wu; Gal, Yarin and Srivastava, Akash: “Fast and Scalable Bayesian Deep Learning by Weight-Perturbation in Adam”. In: *International Conference on Machine Learning* (2018) (cit. on pp. 62, 63).

[Kha23] Khan, Mohammad Emtiyaz and Rue, Håvard: “The Bayesian Learning Rule”. In: *Journal of Machine Learning Research* (2023) (cit. on p. 17).

[Kim18] Kim, Taesup; Yoon, Jaesik; Dia, Ousmane; Kim, Sungwoong; Bengio, Yoshua and Ahn, Sungjin: “Bayesian Model-Agnostic Meta-Learning”. In: *Advances in Neural Information Processing Systems* (2018) (cit. on pp. 40, 44, 61, 93).

[Kim19] Kim, Hyunjik; Mnih, Andriy; Schwarz, Jonathan; Garnelo, Marta; Eslami, S. M. Ali; Rosenbaum, Dan; Vinyals, Oriol and Teh, Yee Whye: “Attentive Neural Processes”. In: *International Conference on Learning Representations* (2019) (cit. on pp. 35, 38, 44–47, 51, 60, 61, 68, 71, 127–129, 132, 133, 148, 149, 151).

[Kim24] Kim, Minyoung and Hospedales, Timothy: “A Hierarchical Bayesian Model for Deep Few-Shot Meta Learning”. In: *International Conference on Learning Representations* (2024) (cit. on pp. 40, 93).

[Kin13] Kingma, Diederik P. and Welling, Max: "Auto-Encoding Variational Bayes". In: *International Conference on Learning Representations* (2013) (cit. on pp. 1, 3, 17, 34, 44, 48, 60–62, 65, 147).

[Kin15] Kingma, Diederik P. and Ba, Jimmy: "Adam: A Method for Stochastic Optimization". In: *International Conference on Learning Representations* (2015) (cit. on pp. 13, 17, 37, 51, 67, 146).

[Kin16] Kingma, Durk P; Salimans, Tim; Jozefowicz, Rafal; Chen, Xi; Sutskever, Ilya and Welling, Max: "Improved Variational Inference with Inverse Autoregressive Flow". In: *Advances in Neural Information Processing Systems* (2016) (cit. on p. 19).

[Kin19] Kingma, Diederik P. and Welling, Max: "An Introduction to Variational Autoencoders". In: *Foundations and Trends in Machine Learning* (2019) (cit. on pp. 17, 18, 20, 92).

[Kin78] Kingman, J. F. C.: "Uses of Exchangeability". In: *The Annals of Probability* (1978) (cit. on pp. 6, 8).

[Kle12] Kleijn, B.J.K. and Vaart, A.W. van der: "The Bernstein-Von-Mises theorem under misspecification". In: *Electronic Journal of Statistics* (2012) (cit. on pp. 9, 12).

[Koc15] Koch, Gregory; Zemel, Richard and Salakhutdinov, Ruslan: "Siamese Neural Networks for One-shot Image Recognition". In: *International Conference on Machine Learning* (2015) (cit. on pp. 44, 61).

[Kol09] Koller, D. and Friedman, N.: Probabilistic Graphical Models: Principles and Techniques. MIT Press, 2009 (cit. on pp. 7, 8, 62).

[Kri12] Krizhevsky, Alex; Sutskever, Ilya and Hinton, Geoffrey E: "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems* (2012) (cit. on pp. 1, 61).

[Kri20] Krippendorf, Sven and Syvaeri, Marc: "Detecting symmetries with neural networks". In: *Machine Learning: Science and Technology* (2020) (cit. on p. 94).

[Kuc17] Kucukelbir, Alp; Blei, David M.; Gelman, Andrew; Ranganath, Rajesh and Tran, Dustin: "Automatic Differentiation Variational Inference". In: *Journal of Machine Learning Research* (2017) (cit. on p. 17).

[Kul18] Kuleshov, Volodymyr; Fenner, Nathan and Ermon, Stefano: "Accurate Uncertainties for Deep Learning Using Calibrated Regression". In: *International Conference on Machine Learning* (2018) (cit. on p. 93).

[Kul51] Kullback, S. and Leibler, R. A.: "On Information and Sufficiency". In: *The Annals of Mathematical Statistics* (1951) (cit. on pp. 9, 62).

[Kus64] Kushner, H. J.: "A New Method of Locating the Maximum Point of an Arbitrary Multipeak Curve in the Presence of Noise". In: *Journal of Basic Engineering* (1964) (cit. on pp. 21, 79).

[Lak11] Lake, Brenden M.; Salakhutdinov, Ruslan; Gross, Jason and Tenenbaum, Joshua B.: "One Shot Learning of Simple Visual Concepts". In: *Cognitive Science* (2011) (cit. on p. 44).

[Lak16] Lake, Brenden M.; Ullman, Tomer D.; Tenenbaum, Joshua B. and Gershman, Samuel J.: "Building Machines That Learn and Think Like People". In: *Behavioral and Brain Sciences* (2016) (cit. on p. 77).

[Lap10] Laplace, P.S. de: Mémoire sur les approximations des formules qui sont fonctions de très-grands nombres, et sur leur application aux probabilités. Memoires de l'Academie des Sciences des Paris, 1810 (cit. on p. 9).

[Law02] Lawrence, Neil; Seeger, Matthias and Herbrich, Ralf: "Fast Sparse Gaussian Process Methods: The Informative Vector Machine". In: *Advances in Neural Information Processing Systems* (2002) (cit. on p. 43).

[Laz10] Lazaro-Gredilla, M. and Figueiras-Vidal, A. R.: "Marginalized Neural Network Mixtures for Large-Scale Regression". In: *IEEE Transactions on Neural Networks* (2010) (cit. on p. 43).

[Le18] Le, Tuan Anh; Kim, Hyunjik and Garnelo, Marta: "Empirical Evaluation of Neural Process Objectives". In: *NeurIPS Workshop on Bayesian Deep Learning* (2018) (cit. on pp. 36, 38, 46, 92).

[LeC10] LeCun, Yann; Cortes, Corinna and Burges, Christopher J.C.: MNIST Handwritten Digit Database. 2010. url: http://yann.lecun.com/exdb/mnist/ (cit. on pp. 72, 132, 155).

[LeC15] LeCun, Yann; Bengio, Yoshua and Hinton, Geoffrey: "Deep learning". In: *Nature* (2015) (cit. on pp. 1, 13).

[LeC89] LeCun, Y.; Boser, B.; Denker, J. S.; Henderson, D.; Howard, R. E.; Hubbard, W. and Jackel, L. D.: "Backpropagation Applied to Handwritten Zip Code Recognition". In: *Neural Computation* (1989) (cit. on p. 1).

[Lee20] Lee, Juho; Lee, Yoonho; Kim, Jungtaek; Yang, Eunho; Hwang, Sung Ju and Teh, Yee Whye: "Bootstrapping Neural Processes". In: *Advances in Neural Information Processing Systems* (2020) (cit. on pp. 38, 61, 68, 148, 149, 151).

[Lei20] Leibniz, G.W.F. von; Child, J.M. and Gerhardt, C.I.: The Early Mathematical Manuscripts of Leibniz: Translated from the Latin Texts Published by Carl Immanuel Gerhardt with Critical and Historical Notes. Open Court Publishing Company, 1920 (cit. on p. 13).

[Len08] Lengerer, Wolfgang: "Sauerstoffanreicherung durch Druckwechseladsorption für Membran-Brennstoffzellensysteme". PhD thesis. University of Stuttgart, 2008 (cit. on pp. v, vii).

[Li16] Li, Ke and Malik, Jitendra: "Learning to Optimize". In: *International Conference on Machine Learning* (2016) (cit. on pp. 38, 44, 77).

[Li17a] Li, Ke and Malik, Jitendra: “Learning to Optimize Neural Nets”. In: *arXiv* (2017) (cit. on pp. 38, 77).

[Li17b] Li, Lisha; Jamieson, Kevin; DeSalvo, Giulia; Rostamizadeh, Afshin and Talwalkar, Ameet: “Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization”. In: *Journal of Machine Learning Research* (2017) (cit. on p. 133).

[Li17c] Li, Zhenguo; Zhou, Fengwei; Chen, Fei and Li, Hang: “Meta-SGD: Learning to Learn Quickly for Few-Shot Learning”. In: *arXiv* (2017) (cit. on p. 39).

[Li21] Li, Zongyi; Kovachki, Nikola; Azizzadenesheli, Kamyar; Liu, Burigede; Bhattacharya, Kaushik; Stuart, Andrew and Anandkumar, Anima: “Fourier Neural Operator for Parametric Partial Differential Equations”. In: *International Conference on Learning Representations* (2021) (cit. on p. 94).

[Li23] Li, Ge; Jin, Zeqi; Volpp, Michael; Otto, Fabian; Lioutikov, Rudolf and Neumann, Gerhard: “ProDMP: A Unified Perspective on Dynamic and Probabilistic Movement Primitives”. In: *IEEE Robotics and Automation Letters* (2023) (cit. on p. 3).

[Lin19] Lin, Wu; Khan, Mohammad Emtiyaz and Schmidt, Mark: “Stein’s Lemma for the Reparameterization Trick with Exponential Family Mixtures”. In: *arXiv* (2019) (cit. on pp. 62, 63).

[Lin20] Lin, Wu; Schmidt, Mark and Khan, Mohammad: “Handling the Positive-Definite Constraint in the Bayesian Learning Rule”. In: *International Conference on Machine Learning* (2020) (cit. on pp. 19, 60–63, 70, 150, 155, 156).

[Lin76] Linnainmaa, Seppo: “Taylor Expansion of the Accumulated Rounding Error”. In: *BIT Computer Science and Numerical Mathematics* (1976) (cit. on p. 13).

[Lou17] Louizos, Christos and Welling, Max: “Multiplicative Normalizing Flows for Variational Bayesian Neural Networks”. In: *International Conference on Machine Learning* (2017) (cit. on p. 43).

[Lou19] Louizos, Christos; Shi, Xiahan; Schutte, Klamer and Welling, M.: “The Functional Neural Process”. In: *Advances in Neural Information Processing Systems* (2019) (cit. on pp. 38, 44, 61).

[Maa16] Maaløe, Lars; Sønderby, Casper Kaae; Sønderby, Søren Kaae and Winther, Ole: “Auxiliary Deep Generative Models”. In: *International Conference on Machine Learning* (2016) (cit. on p. 18).

[Mac03] MacKay, David J. C.: Information Theory, Inference, and Learning Algorithms. Cambridge University Press, 2003 (cit. on pp. 3, 5, 7, 9, 12, 14–16, 32, 61, 63, 65, 93).

[Mac92a] MacKay, David J. C.: “A Practical Bayesian Framework for Backpropagation Networks”. In: *Neural Computation* (1992) (cit. on pp. 43, 128).

[Mac92b] MacKay, David J. C.: “The Evidence Framework Applied to Classification Networks”. In: *Neural Computation* (1992) (cit. on p. 28).

[Mar15] Martín Abadi et al.: TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. 2015. url: https://www.tensorflow.org/ (cit. on pp. 17, 22, 66).

[Mar21] Markou, Stratis; Requeima, James; Bruinsma, Wessel and Turner, Richard: "Efficient Gaussian Neural Processes for Regression". In: *ICML Workshop on Uncertainty and Robustness in Deep Learning* (2021) (cit. on pp. 38, 93).

[Mar22] Markou, Stratis; Requeima, James; Bruinsma, Wessel; Vaughan, Anna and Turner, Richard E: "Practical Conditional Neural Process Via Tractable Dependent Predictions". In: *International Conference on Learning Representations* (2022) (cit. on pp. 38, 93).

[Mar23] Maraval, Alexandre; Zimmer, Matthieu; Grosnit, Antoine and Ammar, Haitham Bou: "End-to-End Meta-Bayesian Optimisation with Transformer Neural Processes". In: *Advances in Neural Information Processing Systems* (2023) (cit. on p. 94).

[Mar84] Markov, A.: "On certain applications of algebraic continued fractions". PhD thesis. St. Petersburg, 1884 (cit. on p. 33).

[Meg22] Megerle, Denis: "Stable Optimization of Gaussian Likelihoods". Master thesis. Karlsruhe Institue of Technology, Karlsruhe, Germany, 2022 (cit. on p. 4).

[Met19] Metz, Luke; Maheswaranathan, Niru; Cheung, Brian and Sohl-Dickstein, Jascha: "Learning Unsupervised Learning Rules". In: *International Conference on Learning Representations* (2019) (cit. on p. 77).

[Met53] Metropolis, Nicholas; Rosenbluth, Arianna W.; Rosenbluth, Marshall N.; Teller, Augusta H. and Teller, Edward: "Equation of State Calculations by Fast Computing Machines". In: *The Journal of Chemical Physics* (1953) (cit. on p. 16).

[Mni15a] Mnih, Volodymyr et al.: "Human-level control through deep reinforcement learning". In: *Nature* (2015) (cit. on p. 1).

[Mni15b] Mnih, Volodymyr et al.: "Human-level control through deep reinforcement learning". In: *Nature* (2015) (cit. on p. 60).

[Moc75] Mockus, J.: "On Bayesian Methods for Seeking the Extremum". In: *Optimization Techniques IFIP Technical Conference* (1975) (cit. on pp. 21, 76, 79).

[Mur23] Murphy, Kevin P.: Probabilistic Machine Learning: Advanced Topics. MIT Press, 2023 (cit. on pp. 5, 10, 13, 14, 17, 18, 23).

[Nad20] Naderiparizi, Saeid; Chiu, Kenny; Bloem-Reddy, Benjamin and Wood, Frank: "Uncertainty in Neural Processes". In: *arXiv* (2020) (cit. on p. 35).

[Nea01] Neal, Radford M.: "Annealed importance sampling". In: *Statistics and Computing* (2001) (cit. on pp. 15, 93).

[Nea93] Neal, Radford M: Probabilistic inference using Markov chain Monte Carlo methods. Technical Report. 1993 (cit. on p. 16).

[Nea96] Neal, Radford M.: Bayesian Learning for Neural Networks. Springer, 1996 (cit. on pp. 13, 43, 62).

[Nea98] Neal, R. M. and Hinton, G. E.: "A new view of the EM algorithm that justifies incremental, sparse and other variants". In: *Learning in Graphical Models* (1998) (cit. on p. 19).

[Ngu22] Nguyen, Tung and Grover, Aditya: "Transformer Neural Processes: Uncertainty-Aware Meta Learning Via Sequence Modeling". In: *International Conference on Machine Learning* (2022) (cit. on p. 38).

[Ngu23] Nguyen, Tuan Dung et al.: "AstroLLaMA: Towards Specialized Foundation Models in Astronomy". In: *arXiv* (2023) (cit. on p. 1).

[Nic18] Nichol, Alex; Achiam, Joshua and Schulman, John: "On First-Order Meta-Learning Algorithms". In: *arXiv* (2018) (cit. on pp. 39, 77).

[Øks10] Øksendal, B.: Stochastic Differential Equations: An Introduction with Applications. Springer, 2010 (cit. on p. 8).

[Ope24] OpenAI et al.: "GPT-4 Technical Report". In: *arXiv* (2024) (cit. on p. 1).

[Osw23] Oswald, Johannes von; Niklasson, Eyvind; Randazzo, Ettore; Sacramento, João; Mordvintsev, Alexander; Zhmoginov, Andrey and Vladymyrov, Max: "Transformers learn in-context by gradient descent". In: *International Conference on Machine Learning* (2023) (cit. on p. 1).

[Pap24] Papamarkou, Theodore et al.: "Position: Bayesian Deep Learning is Needed in the Age of Large-Scale AI". In: (2024) (cit. on pp. 2, 7, 11).

[Par18] Parmar, Niki; Vaswani, Ashish; Uszkoreit, Jakob; Kaiser, Lukasz; Shazeer, Noam M.; Ku, Alexander and Tran, Dustin: "Image Transformer". In: *International Conference on Machine Learning* (2018) (cit. on p. 60).

[Par98] Parisi, G.: Statistical Field Theory. Avalon Publishing, 1998 (cit. on p. 18).

[Pas19] Paszke, Adam et al.: "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems* (2019) (cit. on pp. 17, 22, 66).

[Pat21] Patterson, David; Gonzalez, Joseph; Le, Quoc; Liang, Chen; Munguia, Lluis-Miquel; Rothchild, Daniel; So, David; Texier, Maud and Dean, Jeff: "Carbon Emissions and Large Neural Network Training". In: *arXiv* (2021) (cit. on pp. 2, 94).

[Per18] Perrone, Valerio; Jenatton, Rodolphe; Seeger, Matthias W and Archambeau, Cedric: "Scalable Hyperparameter Transfer Learning". In: *Advances in Neural Information Processing Systems* (2018) (cit. on pp. 44, 53, 78, 131).

[Pet10] Peters, Jan; Mülling, Katharina and Altün, Yasemin: "Relative Entropy Policy Search". In: *AAAI Conference on Artificial Intelligence* (2010) (cit. on p. 17).

[Pic13] Picheny, Victor; Wagner, Tobias and Ginsbourger, David: “A Benchmark of Kriging-based Infill Criteria for Noisy Optimization”. In: *Structural and Multidisciplinary Optimization* (2013) (cit. on pp. 71, 84).

[Pol20] Pol, Elise van der; Worrall, Daniel E.; Hoof, Herke van; Oliehoek, Frans A. and Welling, Max: “MDP Homomorphic Networks: Group Symmetries in Reinforcement Learning”. In: *Advances in Neural Information Processing Systems* (2020) (cit. on p. 94).

[Pre07] Press, W.H.; Teukolsky, S.A.; Vetterling, W.T. and Flannery, B.P.: Numerical Recipes: The Art of Scientific Computing. Cambridge University Press, 2007 (cit. on pp. 1, 15).

[Qui05] Quiñonero-Candela, Joaquin and Rasmussen, Carl Edward: “A Unifying View of Sparse Approximate Gaussian Process Regression”. In: *Journal of Machine Learning Research* (2005) (cit. on p. 43).

[Rad19] Radford, Alec; Wu, Jeff; Child, Rewon; Luan, David; Amodei, Dario and Sutskever, Ilya: “Language Models are Unsupervised Multitask Learners”. In: *arXiv* (2019) (cit. on pp. 1, 60).

[Rad21] Radford, Alec et al.: “Learning Transferable Visual Models From Natural Language Supervision”. In: *International Conference on Machine Learning* (2021) (cit. on p. 1).

[Raf20] Raffel, Colin; Shazeer, Noam; Roberts, Adam; Lee, Katherine; Narang, Sharan; Matena, Michael; Zhou, Yanqi; Li, Wei and Liu, Peter J.: “Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer”. In: *Journal of Machine Learning Research* (2020) (cit. on p. 1).

[Rai19] Raissi, Maziar; Perdikaris, Paris and Karniadakis, George E: “Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations”. In: *Journal of Computational Physics* (2019) (cit. on p. 93).

[Raj19] Rajeswaran, Aravind; Finn, Chelsea; Kakade, Sham M and Levine, Sergey: “Meta-Learning with Implicit Gradients”. In: *Advances in Neural Information Processing Systems* (2019) (cit. on p. 39).

[Ram21] Ramesh, Aditya; Pavlov, Mikhail; Goh, Gabriel; Gray, Scott; Voss, Chelsea; Radford, Alec; Chen, Mark and Sutskever, Ilya: “Zero-Shot Text-to-Image Generation”. In: *arXiv* (2021) (cit. on p. 1).

[Ran14] Ranganath, Rajesh; Gerrish, Sean and Blei, David: “Black Box Variational Inference”. In: *International Conference on Artificial Intelligence and Statistics* (2014) (cit. on pp. 17, 62).

[Ran16] Ranganath, Rajesh; Tran, Dustin and Blei, David: “Hierarchical Variational Models”. In: *International Conference on Machine Learning* (2016) (cit. on p. 18).

[Ras05] Rasmussen, Carl Edward and Williams, Christopher K. I.: Gaussian Processes for Machine Learning. MIT Press, 2005 (cit. on pp. 21, 43, 79, 128).

[Rav17] Ravi, Sachin and Larochelle, Hugo: “Optimization as a Model for Few-Shot Learning”. In: *International Conference on Learning Representations* (2017) (cit. on pp. 38, 44, 61).

[Rav19] Ravi, Sachin and Beatson, Alex: “Amortized Bayesian Meta-Learning”. In: *International Conference on Learning Representations* (2019) (cit. on pp. 40, 93).

[Rez14] Rezende, Danilo Jimenez; Mohamed, Shakir and Wierstra, Daan: “Stochastic Backpropagation and Approximate Inference in Deep Generative Models”. In: *International Conference on Machine Learning* (2014) (cit. on pp. 17, 44, 61).

[Rez15] Rezende, Danilo and Mohamed, Shakir: “Variational Inference with Normalizing Flows”. In: *International Conference on Machine Learning* (2015) (cit. on pp. 1, 19).

[Ric06] Rice, John A.: Mathematical Statistics and Data Analysis. Duxbury Press, 2006 (cit. on pp. 6, 9).

[Ric23] Ricci, Matt; Moriel, Noa; Piran, Zoe and Nitzan, Mor: “Phase2vec: dynamical systems embedding with a physics-informed convolutional network”. In: *International Conference on Learning Representations* (2023) (cit. on p. 94).

[Rob07] Robert, Christian P.: The Bayesian Choice: From Decision-Theoretic Foundations to Computational Implementation. Springer, 2007 (cit. on pp. 13, 14).

[Rob51] Robbins, H. and Monro, S.: “A stochastic approximation method”. In: *Annals of Mathematical Statistics* (1951) (cit. on pp. 13, 17, 37).

[Rom22] Rombach, Robin; Blattmann, Andreas; Lorenz, Dominik; Esser, Patrick and Ommer, Björn: “High-resolution image synthesis with latent diffusion models”. In: *IEEE Conference on Computer Vision and Pattern Recognition* (2022) (cit. on p. 1).

[Ros20] Roscher, Ribana; Bohn, Bastian; Duarte, Marco F and Garcke, Jochen: “Explainable machine learning for scientific insights and discoveries”. In: *arXiv* (2020) (cit. on p. 93).

[Ros62] Rosenblatt, F.: Principles of Neurodynamics. Spartan Books, 1962 (cit. on p. 1).

[Rum86] Rumelhart, David E.; Hinton, Geoffrey E. and Williams, Ronald J.: “Learning representations by back-propagating errors”. In: *Nature* (1986) (cit. on pp. 1, 13, 17, 37).

[Run00] Runarsson, T. P. and Jonsson, M. T.: “Evolution and Design of Distributed Learning Rules”. In: *IEEE Symposium on Combinations of Evolutionary Computation and Neural Networks* (2000) (cit. on p. 77).

[Rus18a] Russo, Daniel J.; Van Roy, Benjamin; Kazerouni, Abbas; Osband, Ian and Wen, Zheng: “A Tutorial on Thompson Sampling”. In: *Foundations and Trends in Machine Learning* (2018) (cit. on p. 71).

[Rus18b] Rusu, Andrei A.; Rao, Dushyant; Sygnowski, Jakub; Vinyals, Oriol; Pascanu, Razvan; Osindero, Simon and Hadsell, Raia: “Meta-Learning with Latent Embedding Optimization”. In: *International Conference on Learning Representations* (2018) (cit. on pp. 39, 44).

[San16] Santoro, Adam; Bartunov, Sergey; Botvinick, Matthew M.; Wierstra, Daan and Lillicrap, Timothy P.: "Meta-Learning with Memory-Augmented Neural Networks". In: *International Conference on Machine Learning* (2016) (cit. on pp. 44, 61).

[Sau95] Saul, Lawrence and Jordan, Michael: "Exploiting Tractable Substructures in Intractable Networks". In: *Advances in Neural Information Processing Systems* (1995) (cit. on p. 18).

[Sch15] Schulman, John; Levine, Sergey; Abbeel, Pieter; Jordan, Michael I. and Moritz, Philipp: "Trust Region Policy Optimization". In: *International Conference on Machine Learning* (2015) (cit. on p. 17).

[Sch16a] Schilling, Nicolas; Wistuba, Martin and Schmidt-Thieme, Lars: "Scalable Hyperparameter Optimization with Products of Gaussian Process Experts". In: *European Conference on Machine Learning and Knowledge Discovery in Databases* (2016) (cit. on p. 78).

[Sch16b] Schulman, John; Moritz, Philipp; Levine, Sergey; Jordan, Michael I. and Abbeel, Pieter: "High-Dimensional Continuous Control Using Generalized Advantage Estimation". In: *International Conference on Learning Representations* (2016) (cit. on pp. 174, 176).

[Sch17] Schulman, John; Wolski, Filip; Dhariwal, Prafulla; Radford, Alec and Klimov, Oleg: "Proximal Policy Optimization Algorithms". In: *arXiv* (2017) (cit. on pp. 82, 174, 175).

[Sch21] Schuhmann, Christoph; Vencu, Richard; Beaumont, Romain; Kaczmarczyk, Robert; Mullis, Clayton; Katta, Aarush; Coombes, Theo; Jitsev, Jenia and Komatsuzaki, Aran: "LAION-400M: Open Dataset of CLIP-Filtered 400 Million Image-Text Pairs". In: *arXiv* (2021) (cit. on p. 1).

[Sch50] Schwartz, L.: Théorie des distributions. Hermann, 1950 (cit. on p. 10).

[Sch87] Schmidhuber, Jürgen: "Evolutionary Principles in Self-Referential Learning. On Learning how to Learn: The Meta-Meta-Meta...-Hook". Diploma thesis. Technical University of Munich, Munich, Germany, 1987 (cit. on pp. 2, 38, 44, 61, 77).

[Sch92] Schmidhuber, Jürgen: "Learning to Control Fast-Weight Memories: An Alternative to Dynamic Recurrent Networks". In: *Neural Computation* (1992) (cit. on pp. 38, 44, 61).

[Sel23] Seligmann, Florian; Becker, Philipp; Volpp, Michael and Neumann, Gerhard: "Beyond Deep Ensembles: A Large-Scale Evaluation of Bayesian Deep Learning under Distribution Shift". In: *Advances in Neural Information Processing Systems* (2023) (cit. on pp. 4, 92, 93).

[Sha16] Shahriari, Bobak; Swersky, Kevin; Wang, Ziyu; Adams, Ryan P. and Freitas, Nando de: "Taking the Human Out of the Loop: A Review of Bayesian Optimization". In: *Proceedings of the IEEE* (2016) (cit. on pp. 2, 3, 20, 33, 60, 71, 76, 79).

[Shi17] Shilton, Alistair; Gupta, Sunil; Rana, Santu and Venkatesh, Svetha: “Regret Bounds for Transfer Learning in Bayesian Optimisation”. In: *International Conference on Artificial Intelligence and Statistics* (2017) (cit. on p. 77).

[Sil17] Silver, David et al.: “Mastering the game of Go without human knowledge”. In: *Nature* (2017) (cit. on p. 1).

[Sim15] Simonyan, Karen and Zisserman, Andrew: “Very Deep Convolutional Networks for Large-Scale Image Recognition.” In: *International Conference on Learning Representations* (2015) (cit. on p. 1).

[Smo01] Smola, Alex J. and Bartlett, Peter L.: “Sparse Greedy Gaussian Process Regression”. In: *Advances in Neural Information Processing Systems* (2001) (cit. on p. 43).

[Sne05] Snelson, Edward and Ghahramani, Zoubin: “Sparse Gaussian Processes Using Pseudo-Inputs”. In: *Advances in Neural Information Processing Systems* (2005) (cit. on p. 43).

[Sne17] Snell, Jake; Swersky, Kevin and Zemel, Richard S.: “Prototypical Networks for Few-shot Learning”. In: *Advances in Neural Information Processing Systems* (2017) (cit. on pp. 44, 61).

[Sno12] Snoek, Jasper; Larochelle, Hugo and Adams, Ryan P.: “Practical Bayesian Optimization of Machine Learning Algorithms”. In: *Advances in Neural Information Processing Systems* (2012) (cit. on pp. 20, 43, 76, 82).

[Sob67] Sobol’, Il’ya Meerovich: “On the distribution of points in a cube and the approximate evaluation of integrals”. In: *Zhurnal Vychislitel’noi Matematiki i Matematicheskoi Fiziki* (1967) (cit. on p. 82).

[Soh15a] Sohl-Dickstein, Jascha; Weiss, Eric; Maheswaranathan, Niru and Ganguli, Surya: “Deep Unsupervised Learning using Nonequilibrium Thermodynamics”. In: *International Conference on Machine Learning* (2015) (cit. on p. 1).

[Soh15b] Sohn, Kihyuk; Lee, Honglak and Yan, Xinchen: “Learning Structured Output Representation using Deep Conditional Generative Models”. In: *Advances in Neural Information Processing Systems* (2015) (cit. on p. 44).

[Spr16] Springenberg, Jost Tobias; Klein, Aaron; Falkner, Stefan and Hutter, Frank: “Bayesian Optimization with Robust Bayesian Neural Networks”. In: *Advances in Neural Information Processing Systems* (2016) (cit. on pp. 20, 21, 78).

[Sri10] Srinivas, Niranjan; Krause, Andreas; Kakade, Sham and Seeger, Matthias W.: “Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design”. In: *International Conference on Machine Learning* (2010) (cit. on pp. 21, 79, 81).

[Str19] Strubell, Emma; Ganesh, Ananya and McCallum, Andrew: “Energy and Policy Considerations for Deep Learning in NLP”. In: *arXiv* (2019) (cit. on pp. 2, 94).

[Sun17] Sung, Flood; Yang, Yongxin; Zhang, Li; Xiang, Tao; Torr, Philip H. S. and Hospedales, Timothy M.: "Learning to Compare: Relation Network for Few-Shot Learning". In: *IEEE Conference on Computer Vision and Pattern Recognition* (2017) (cit. on p. 44).

[Sut18] Sutton, Richard S. and Barto, Andrew G.: Reinforcement Learning: An Introduction. MIT Press, 2018 (cit. on pp. 4, 20, 75, 80).

[Sut19] Sutton, Richard: The Bitter Lesson. 2019. url: http://www.incompleteideas.net/IncIdeas/BitterLesson.html (cit. on pp. 1, 94).

[Swe13] Swersky, Kevin; Snoek, Jasper and Adams, Ryan P: "Multi-Task Bayesian Optimization". In: *Advances in Neural Information Processing Systems* (2013) (cit. on pp. 76, 77).

[Sze78] Szego, G. P. and Dixon, L. C. W.: Towards global optimisation. North-Holland Publishing Company, 1978 (cit. on p. 71).

[Tai23] Tailor, Dharmesh; Khan, Mohammad Emtiyaz and Nalisnick, Eric T.: "Exploiting Inferential Structure in Neural Processes". In: *Uncertainty in Artificial Intelligence* (2023) (cit. on p. 92).

[The16] Theckel Joy, Tinu; Rana, Santu; Gupta, Sunil and Venkatesh, Svetha: "Flexible Transfer Learning Framework for Bayesian Optimisation". In: *Advances in Knowledge Discovery and Data Mining* (2016) (cit. on p. 77).

[Tho33] Thompson, William R: "On the Likelihood That One Unknown Probability Exceeds Another in View Of The Evidence Of Two Samples". In: *Biometrika* (1933) (cit. on p. 21).

[Thr98] Thrun, Sebastian and Pratt, Lorien: Learning to Learn. Kluwer Academic Publishers, 1998 (cit. on pp. 2, 44, 61, 77).

[Tou23] Touvron, Hugo et al.: "LLaMA: Open and Efficient Foundation Language Models". In: *arXiv* (2023) (cit. on p. 1).

[Tur11] Turner, R. E. and Sahani, M.: "Two problems with variational expectation maximisation for time-series models". In: *Bayesian Time Series Models* (2011) (cit. on pp. 18, 20).

[Uda24] Udandarao, Vishaal; Prabhu, Ameya; Ghosh, Adhiraj; Sharma, Yash; Torr, Philip H. S.; Bibi, Adel; Albanie, Samuel and Bethge, Matthias: "No SZero-Shot"Without Exponential Data: Pretraining Concept Frequency Determines Multimodal Model Performance". In: *arXiv* (2024) (cit. on p. 94).

[Udr20] Udrescu, Silviu-Marian and Tegmark, Max: "AI Feynman: A physics-inspired method for symbolic regression". In: *Science Advances* (2020) (cit. on p. 93).

[Vaa00] Vaart, A.W. van der: Asymptotic Statistics. Cambridge University Press, 2000 (cit. on p. 9).

[Vad21] Vadyala, Shashank; Betgeri, Sai Nethra; Matthews, John and Matthews, Elizabeth: “A Review of Physics-based Machine Learning in Civil Engineering”. In: *Results in Engineering* (2021) (cit. on p. 93).

[Val18] Valkov, Lazar; Jenatton, Rodolphe; Winkelmolen, Fela and Archambeau, Cédric: “A simple transfer-learning extension of Hyperband”. In: *NeurIPS Workshop on Meta-Learning* (2018) (cit. on p. 89).

[Van19] Vanschoren, Joaquin: “Meta-Learning”. In: *Automated Machine Learning: Methods, Systems, Challenges* (2019) (cit. on p. 37).

[van20] van der Wilk, Mark; Dutordoir, Vincent; John, ST; Artemev, Artem; Adam, Vincent and Hensman, James: “A Framework for Interdomain and Multioutput Gaussian Processes”. In: *arXiv* (2020) (cit. on p. 22).

[Vas17] Vaswani, Ashish; Shazeer, Noam; Parmar, Niki; Uszkoreit, Jakob; Jones, Llion; Gomez, Aidan N; Kaiser, Łukasz and Polosukhin, Illia: “Attention is All you Need”. In: *Advances in Neural Information Processing Systems* (2017) (cit. on pp. 1, 23, 94, 127).

[Vas21] Vashishtha, Anushka: “Data Augmentation for Neural Processes”. Master thesis. Technical University Munich, Munich, Germany, 2021 (cit. on p. 4).

[Vil05] Vilalta, Ricardo and Drissi, Youssef: “A Perspective View and Survey of Meta-Learning”. In: *Artificial Intelligence Review* (2005) (cit. on pp. 2, 44, 61).

[Vin16] Vinyals, Oriol; Blundell, Charles; Lillicrap, Timothy; Kavukcuoglu, Koray and Wierstra, Daan: “Matching Networks for One Shot Learning”. In: *Advances in Neural Information Processing Systems* (2016) (cit. on pp. 44, 61).

[Vin17] Vinyals, Oriol: Model vs Optimization Meta Learning. 2017. url: https://evolution.ml/pdf/vinyals.pdf (cit. on pp. 37, 38).

[Vol17] Volpp, Michael: “Running of Radiative Neutrino Masses - A Study of the Zee-Babu Model”. Master thesis. Max Planck Institute for Physics, Munich, Germany, 2017 (cit. on p. 1).

[Vol20] Volpp, Michael; Fröhlich, Lukas P.; Fischer, Kirsten; Doerr, Andreas; Falkner, Stefan; Hutter, Frank and Daniel, Christian: “Meta-Learning Acquisition Functions for Transfer Learning in Bayesian Optimization”. In: *International Conference on Learning Representations* (2020) (cit. on pp. 4, 44, 71, 76).

[Vol21] Volpp, Michael; Flürenbrock, Fabian; Großberger, Lukas; Daniel, Christian and Neumann, Gerhard: “Bayesian Context Aggregation for Neural Processes”. In: *International Conference on Learning Representations* (2021) (cit. on pp. 3, 42, 60, 61, 68, 72, 148, 152).

[Vol23] Volpp, Michael; Dahlinger, Philipp; Becker, Philipp; Daniel, Christian and Neumann, Gerhard: “Accurate Bayesian Meta-Learning by Accurate Task Posterior Inference”. In: *International Conference on Learning Representations* (2023) (cit. on pp. 3, 60).

[Wag19] Wagstaff, Edward; Fuchs, Fabian B.; Engelcke, Martin; Posner, Ingmar and Osborne, Michael A.: "On the Limitations of Representing Functions on Sets". In: *International Conference on Machine Learning* (2019) (cit. on pp. 34, 41, 44, 61, 64).

[Wai08] Wainwright, Martin J. and Jordan, Michael I.: "Graphical Models, Exponential Families, and Variational Inference". In: *Foundations and Trends in Machine Learning* (2008) (cit. on p. 18).

[Wan20a] Wang, Qi and Van Hoof, Herke: "Doubly Stochastic Variational Inference for Neural Processes with Hierarchical Latent Variables". In: *International Conference on Machine Learning* (2020) (cit. on pp. 38, 61).

[Wan20b] Wang, Yaqing; Yao, Quanming; Kwok, James T. and Ni, Lionel M.: "Generalizing from a Few Examples: A Survey on Few-shot Learning". In: *ACM Computing Surveys* (2020) (cit. on p. 37).

[Wan21] Wang, Rui: "Physics-Guided Deep Learning for Dynamical Systems: A survey". In: *arXiv* (2021) (cit. on p. 94).

[Was04] Wasserman, Larry: All of statistics: a concise course in statistical inference. Springer, 2004 (cit. on pp. 5, 6, 9, 10).

[Wei22] Wei, Jason et al.: "Emergent Abilities of Large Language Models". In: *Transactions on Machine Learning Research* (2022) (cit. on p. 1).

[Wil16] Wilson, Andrew Gordon; Hu, Zhiting; Salakhutdinov, Ruslan and Xing, Eric P.: "Deep Kernel Learning". In: *International Conference on Artificial Intelligence and Statistics* (2016) (cit. on p. 43).

[Wil18] Wilson, James T.; Hutter, Frank and Deisenroth, Marc Peter: "Maximizing acquisition functions for Bayesian optimization". In: *arXiv* (2018) (cit. on p. 22).

[Wil20] Wilson, Andrew Gordon and Izmailov, Pavel: "Bayesian Deep Learning and a Probabilistic Perspective of Generalization". In: *Advances in Neural Information Processing Systems* (2020) (cit. on pp. 7, 9, 11–14, 18, 25, 61–63, 92).

[Wil96] Williams, Christopher K. I.: "Computing with Infinite Networks". In: *Advances in Neural Information Processing Systems* (1996) (cit. on p. 43).

[Win05] Winn, J. and Bishop, C.M.: "Variational Message Passing". In: *Journal of Machine Learning Research* (2005) (cit. on p. 62).

[Wis16] Wistuba, Martin; Schilling, Nicolas and Schmidt-Thieme, Lars: "Two-Stage Transfer Surrogate Model for Automatic Hyperparameter Optimization". In: *European Conference on Machine Learning and Knowledge Discovery in Databases* (2016) (cit. on p. 78).

[Wis18] Wistuba, Martin; Schilling, Nicolas and Schmidt-Thieme, Lars: "Scalable Gaussian Process-based Transfer Surrogates for Hyperparameter Optimization". In: *Journal of Machine Learning Research* (2018) (cit. on pp. 76, 78–80, 83, 87, 175).

[Wu17] Wu, Yuhuai; Burda, Yuri; Salakhutdinov, Ruslan and Grosse, Roger B.: “On the Quantitative Analysis of Decoder-Based Generative Models”. In: *International Conference on Learning Representations* (2017) (cit. on p. 93).

[Yog14] Yogatama, Dani and Mann, Gideon: “Efficient Transfer Learning Method for Automatic Hyperparameter Tuning”. In: *International Conference on Artificial Intelligence and Statistics* (2014) (cit. on pp. 77, 128).

[Yu23] Yu, Wenhao et al.: “Language to Rewards for Robotic Skill Synthesis”. In: *arXiv* (2023) (cit. on p. 1).

[Zah17] Zaheer, Manzil; Kottur, Satwik; Ravanbakhsh, Siamak; Póczos, Barnabás; Salakhutdinov, Ruslan and Smola, Alexander J.: “Deep Sets”. In: *Advances in Neural Information Processing Systems* (2017) (cit. on pp. 3, 34, 41, 44, 46, 60, 61, 64).

[Zha20] Zhao, Dominic; Kobayashi, Seijin; Sacramento, João and Oswald, Johannes von: “Meta-Learning via Hypernetworks”. In: *NeurIPS Workshop on Meta-Learning* (2020) (cit. on p. 38).

[Zhu21] Zhuang, Fuzhen; Qi, Zhiyuan; Duan, Keyu; Xi, Dongbo; Zhu, Yongchun; Zhu, Hengshu; Xiong, Hui and He, Qing: “A Comprehensive Survey on Transfer Learning”. In: *Proceedings of the IEEE* (2021) (cit. on pp. 1, 23, 61).

[Zvy23] Zvyagin, Maxim et al.: “GenSLMs: Genome-scale language models reveal SARS-CoV-2 evolutionary dynamics”. In: *International Journal of High Performance Computing Applications* (2023) (cit. on p. 1).

Own Publications

This section contains a complete list of the publications that I have contributed to during my PhD studies.

[1] Doerr, Andreas; Volpp, Michael; Toussaint, Marc; Sebastian, Trimpe and Daniel, Christian: “Trajectory-Based Off-Policy Deep Reinforcement Learning”. In: *International Conference on Machine Learning* (2019).

[2] Volpp, Michael; Fröhlich, Lukas P.; Fischer, Kirsten; Doerr, Andreas; Falkner, Stefan; Hutter, Frank and Daniel, Christian: “Meta-Learning Acquisition Functions for Transfer Learning in Bayesian Optimization”. In: *International Conference on Learning Representations* (2020).

[3] Volpp, Michael; Flürenbrock, Fabian; Großberger, Lukas; Daniel, Christian and Neumann, Gerhard: “Bayesian Context Aggregation for Neural Processes”. In: *International Conference on Learning Representations* (2021).

[4] Gao, Ning; Ziesche, Hanna; Vien, Ngo Anh; Volpp, Michael and Neumann, Gerhard: “What Matters for Meta-Learning Vision Regression Tasks?” In: *IEEE Conference on Computer Vision and Pattern Recognition* (2022).

[5] Arenz, Oleg; Dahlinger, Philipp; Ye, Zihan; Volpp, Michael and Neumann, Gerhard: “A Unified Perspective on Natural Gradient Variational Inference with Gaussian Mixture Models”. In: *Transactions on Machine Learning Research* (2023).

[6] Dahlinger, Philipp; Freymuth, Niklas; Hoang, Tai; Volpp, Michael and Neumann, Gerhard: “Latent Task-Specific Graph Network Simulators”. In: *arXiv* (2023).

[7] Li, Ge; Jin, Zeqi; Volpp, Michael; Otto, Fabian; Lioutikov, Rudolf and Neumann, Gerhard: “ProDMP: A Unified Perspective on Dynamic and Probabilistic Movement Primitives”. In: *IEEE Robotics and Automation Letters* (2023).

[8] Seligmann, Florian; Becker, Philipp; Volpp, Michael and Neumann, Gerhard: “Beyond Deep Ensembles: A Large-Scale Evaluation of Bayesian Deep Learning under Distribution Shift”. In: *Advances in Neural Information Processing Systems* (2023).

[9] Volpp, Michael; Dahlinger, Philipp; Becker, Philipp; Daniel, Christian and Neumann, Gerhard: “Accurate Bayesian Meta-Learning by Accurate Task Posterior Inference”. In: *International Conference on Learning Representations* (2023).

Supervised Student Theses

This section contains a list of student theses that I have supervised during my PhD studies.

[1] Vashishtha, Anushka: “Data Augmentation for Neural Processes”. Master thesis. Technical University Munich, Munich, Germany, 2021.

[2] Florin, Lucas: “Annealed Importance Sampling for Neural Process Training”. Master thesis. Karlsruhe Institute of Technology, Karlsruhe, Germany, 2022.

[3] Megerle, Denis: “Stable Optimization of Gaussian Likelihoods”. Master thesis. Karlsruhe Institue of Technology, Karlsruhe, Germany, 2022.

[4] Seligmann, Florian: “Variational Bayesian Deep Learning for Model Predictive Control”. Bachelor thesis. Karlsruhe Institue of Technology, Karlsruhe, Germany, 2022.

A Bayesian Context Aggregation for Neural Processes

We present the derivation of the Bayesian aggregation update equations (Eqs. (3.10), (3.11)) in more detail. To foster reproducibility, we describe all experimental settings as well as the hyper-parameter optimization procedure used to obtain the results reported in Sec. 3.5, and publish the source code online.[1] We further provide additional experimental results and visualizations of the predictions of the compared architectures.

A.1 Derivation of the Bayesian Aggregation Update Equations

We derive the full Bayesian aggregation update equations without making any factorization assumptions. We start from a Gaussian observation model of the form

$$\begin{aligned} p\left(r_n \mid z\right) &\equiv \mathcal{N}\left(r_n \mid z, \Sigma_{r_n}\right), \\ r_n &= \operatorname{enc}_{r,\phi}\left(x_n^c, y_n^c\right), \\ \Sigma_{r_n} &= \operatorname{enc}_{\Sigma_r,\phi}\left(x_n^c, y_n^c\right), \end{aligned} \tag{A.1}$$

where r_n and Σ_{r_n} are learned by the encoder network. If we impose a Gaussian prior in the latent space, i.e.,

$$p\left(z\right) \equiv \mathcal{N}\left(z \mid \mu_{z,0}, \Sigma_{z,0}\right), \tag{A.2}$$

we arrive at a Gaussian aggregation model which allows to derive the parameters of the posterior distribution, i.e., of

$$q_\phi\left(z \mid \mathcal{D}^c\right) = \mathcal{N}\left(z \mid \mu_z, \Sigma_z\right) \tag{A.3}$$

[1] https://github.com/boschresearch/bayesian-context-aggregation

in closed form using standard Gaussian conditioning [Bis06]:

$$\Sigma_z = \left[\left(\Sigma_{z,0}\right)^{-1} + \sum_{n=1}^{N} \left(\Sigma_{r_n}\right)^{-1} \right]^{-1}, \tag{A.4a}$$

$$\mu_z = \mu_{z,0} + \Sigma_z \sum_{n=1}^{N} \left(\Sigma_{r_n}\right)^{-1} \left(r_n - \mu_{z,0}\right). \tag{A.4b}$$

As the latent space z is shaped by the encoder network, it will find a space where the following factorization assumptions work well (given d_z is large enough):

$$\begin{aligned} \Sigma_{r_n} &= \operatorname{diag}\left(\sigma_{r_n}^2\right), \\ \sigma_{r_n}^2 &= \operatorname{enc}_{\sigma_r^2,\phi}\left(x_n^c, y_n^c\right), \\ \Sigma_{z,0} &= \operatorname{diag}\left(\sigma_{z,0}^2\right). \end{aligned} \tag{A.5}$$

This yields a factorized posterior, i.e.,

$$q_\phi\left(z \mid \mathcal{D}^c\right) = \mathcal{N}\left(z \mid \mu_z, \operatorname{diag}\left(\sigma_z^2\right)\right), \tag{A.6}$$

with

$$\sigma_z^2 = \left[\left(\sigma_{z,0}^2\right)^{\ominus} + \sum_{n=1}^{N} \left(\sigma_{r_n}^2\right)^{\ominus} \right]^{\ominus}, \tag{A.7a}$$

$$\mu_z = \mu_{z,0} + \sigma_z^2 \odot \sum_{n=1}^{N} \left(r_n - \mu_{z,0}\right) \oslash \left(\sigma_{r_n}^2\right). \tag{A.7b}$$

Here $^{\ominus}$, $\odot$ and $\oslash$ denote element-wise inversion, product, and division, respectively. This is the result Eq. (3.10) from the main part of this paper.

A.2 Discussion of VI Likelihood Approximation

To highlight the limitations of the VI approximation, we note that decoder networks of models employing the PB or the MC likelihood approximation are provided with the same context information at training and test time: the latent variable (which is passed on to the decoder in the form of latent samples z (for MC) or in the form of parameters μ_z, σ_z^2 describing the latent distribution (for PB)) is in both cases conditioned only on the context set $\mathcal{D}^c$. In contrast, in the variational approximation Eq. (3.2), the expectation is w.r.t. q_ϕ, conditioned on the union of the context set $\mathcal{D}^c$ and the target set $\mathcal{D}^t$. As $\mathcal{D}^t$ is not available at test time, this introduces a mismatch between how the model is trained and how it is used at test time. Indeed, the decoder is trained on samples from $q_\phi\left(z \mid \mathcal{D}^c \cup \mathcal{D}^t\right)$ but evaluated on samples from $q_\phi\left(z \mid \mathcal{D}^c\right)$. This is not a serious problem when the model is evaluated on context sets with sizes large enough to allow accurate approximations of the true latent posterior distribution. Small context sets, however, usually contain too little information to infer z reliably. Consequently, the distributions $q_\phi\left(z \mid \mathcal{D}^c\right)$ and $q_\phi\left(z \mid \mathcal{D}^c \cup \mathcal{D}^t\right)$ typically differ significantly in this regime. Hence, incentivizing the decoder to yield meaningful predictions on small context sets requires intricate and potentially expensive additional sampling procedures to choose suitable target sets $\mathcal{D}^t$ during training. As a corner case, we point out that it is not possible to train the decoder on samples from the latent prior, because the right hand side of Eq. (3.2) vanishes for $\mathcal{D}^c = \mathcal{D}^t = \emptyset$.

A.3 Self-Attentive Encoder Architectures

Kim et al. [Kim19] propose to use attention-mechanisms to improve the quality of NP-based regression. In general, given a set of key-value pairs $\{(x_n, y_n)\}_{n=1}^N$, $x_n \in \mathbb{R}^{d_x}$, $y_n \in \mathbb{R}^{d_y}$, and a query $x^* \in \mathbb{R}^{d_x}$, an attention mechanism $\mathcal{A}$ produces a weighted sum of the values, with the weights being computed from the keys and the query:

$$\mathcal{A}\left(\{(x_n, y_n)\}_{n=1}^N, x^*\right) = \sum_{n=1}^N w\left(x_n, x^*\right) y_n. \tag{A.8}$$

There are several types of attention mechanisms proposed in the literature [Vas17], each defining a specific form of the weights. *Laplace attention* adjusts the weights according to the spatial distance of keys and query:

$$w_{\mathrm{L}}\left(x_n, x^*\right) \propto \exp\left(-\|x_n - x^*\|_1\right). \tag{A.9}$$

Similarly, *dot-product attention* computes

$$w_{\mathrm{DP}}\left(x_n, x^*\right) \propto \exp\left(x_n^T x^* / \sqrt{d_x}\right). \tag{A.10}$$

A more complex mechanism is *multihead attention*, which employs a set of $3H$ learned linear mappings $\left\{\mathcal{L}_h^K\right\}_{h=1}^H$, $\left\{\mathcal{L}_h^V\right\}_{h=1}^H$, $\left\{\mathcal{L}_h^Q\right\}_{h=1}^H$, where H is a hyperparameter. For each h, these mappings are applied to keys, values, and queries, respectively. Subsequently, dot-product attention is applied to the set of transformed key-value pairs and the transformed query. The resulting H values are then again combined by a further learned linear mapping $\mathcal{L}^O$ to obtain the final result.

Self-attention (SA) is defined by setting the set of queries equal to the set of keys. Therefore, SA produces again a set of N weighted values. Combining SA with an NP-encoder, i.e., applying SA to the set $\{f_x(x_n), r_n\}_{n=1}^N$ of inputs x_n and corresponding latent observations r_n (where we also consider a possible nonlinear transformation f_x of the inputs) and subsequently applying MA yields an interesting baseline for our proposed BA. Indeed, similar to BA, SA computes a weighted sum of the latent observations r_n. Note, however, that SA weighs each latent observation according to some form of spatial relationship of the corresponding input with all other latent observations in the context set. In contrast, BA's weight for a given latent observation is based only on features computed from the context tuple corresponding to this very latent observation and allows to incorporate an estimation of the amount of information contained in the context tuple into the aggregation (cf. Sec. 3.4.1). This leads to several computational advantages of BA over SA: (i) SA scales quadratically in the number N of context tuples, as it has to be evaluated on all N^2 pairs of context tuples. In contrast, BA scales linearly with N. (ii) BA allows for efficient incremental updates when context data arrives sequentially (cf. Eq. (3.11)), while using SA does not provide this possibility: it requires to store and encode the whole context set $\mathcal{D}^c$ at once and to subsequently aggregate the whole set of resulting (SA-weighted) latent observations.

The results in Tabs. 3.6, 3.7, Sec. 3.5 show that multihead SA leads to significant improvements in predictive performance compared to vanilla MA. Therefore, a combination of BA with self-attentive encoders seems promising in situations where computational disadvantages can be accepted in favour of increased predictive performance. Note that BA relies on a second encoder output $\sigma_{r_n}^2$ (in addition to the latent observation r_n) which assesses the information content in each context tuple (x_n, y_n). As each SA-weighted r_n is informed by the other latent observations in the context set, obviously, one would have to also process the set of $\sigma_{r_n}^2$ in a manner consistent with the SA-weighting. We leave such a combination of SA and BA for future research.

A.4 Neural Process-based Models in the Context of Scalable Probabilistic Regression

We discuss in more detail how NP-based models relate to other existing methods for (scalable) probabilistic regression, such as (multi-task) GPs [Ras05, Bar13, Yog14, Gol17], Bayesian neural networks (BNNs) [Mac92a, Gal16], and DeepGPs [Dam13].

NPs are motivated in Garnelo et al. [Gar18b, Gar18c], Kim et al. [Kim19], as well as in our Sec. 3.1, as models which combine the computational efficiency of neural networks with well-calibrated uncertainty estimates (like those of GPs). Indeed, NPs scale linearly in the number N of context and M of target data points, i.e., like $\mathcal{O}(N + M)$, while GPs scale like $\mathcal{O}(N^3 + M^2)$. Furthermore, NPs are shown to exhibit well-calibrated uncertainty estimates. In this sense, NPs can be counted as members of the family of scalable probabilistic regression methods.

A central aspect of NP training which distinguishes NPs from a range of standard methods is that they are trained in a multi-task fashion (cf. Sec. 4.3). This means that NPs rely on data from a set of related source tasks from which they automatically learn powerful priors and the ability to adapt quickly to unseen target tasks. This multi-task training procedure of NPs scales linearly in the number L of source tasks, which makes it possible to train these architectures on large amounts of source data. Applying GPs in such a multi-task setting can be challenging, especially for large numbers of source tasks. Similarly, BNNs as well as DeepGPs are in their vanilla forms specifically designed for the single-task setting. Therefore, GPs, BNNs, and DeepGPs are not directly applicable in the NP multi-task setting, which is why they are typically not considered as baselines for NP-based models, as discussed in [Kim19].

The experiments presented in Garnelo et al. [Gar18b, Gar18c] and Kim et al. [Kim19] focus mainly on evaluating NPs in the context of few-shot probabilistic regression, i.e., on demonstrating the data-efficiency of NPs on the target task after training on data from a range of source tasks. In contrast, the application of NPs in situations with large (> 1000) numbers of context/target points per task has to the best of our knowledge not yet been investigated in detail in the literature. Furthermore, it has not been studied how to apply NPs in situations where only a single or very few source tasks are available. The focus of our paper is a clear-cut comparison of the performance of our BA with traditional MA in the context of NP-based models. Therefore, we also consider experiments similar to those presented in [Gar18b, Gar18c, Kim19] and leave further comparisons with existing methods for (multi-task) probabilistic regressions for future work.

Nevertheless, to illustrate this discussion, we provide two simple GP-based baseline methods: (i) a vanilla GP, which optimizes the hyperparameters on each target task individually and does not use the source data, and (ii) a naive but easily interpretable example of a multi-task GP, which optimizes one set of hyperparameters on all source tasks and uses it for predictions on the target tasks without further adaptation. The results in Tab. A.1 show that those GP-based models can only compete with NPs on function classes where either the inductive bias as given by the kernel functions fits the data well (RBF GP), or on function classes which exhibit a relatively low degree of variability (Quadratic 1D). On more complex function classes, NPs produce predictions of much better quality, as they incorporate the source data more efficiently.

Table A.1: Comparison of the predictive log-likelihood of NP-based architectures with two simple GP-based baselines, (i) Vanilla GP (optimizes the hyperparameters individually on each target task and ignores the source data) (ii) Multitask GP (optimizes one set of hyperparameters on all source tasks and uses them without further adaptation on the target tasks). Both GP implementations use RBF-kernels. As in the main text, we average performance over context sets with sizes $N \in \{0, \dots, 64\}$ for RBF GP and $N \in \{0, \dots, 20\}$ for the other experiments. Multitask GP constitutes the optimal model (assuming it fits the hyperparameters perfectly) for the RBF GP experiment, which explains its superior performance. On the Quadratic 1D experiment, Multitask GP still performs better than the other methods as this function class shows a relatively low degree of variability. In contrast, on more complex experiments like Quadratic 3D and the Furuta dynamics, none of the GP variants is able to produce meaningful results given the small budget of at most 20 context points, while NP-based methods produce predictions of high quality as they incorporate the source data more efficiently.

	NPs with MC-loss		GP	
	BA	MA	Vanilla	Multitask
RBF GP	1.62 ± 0.05	1.07 ± 0.05	1.96	$\mathbf{2.99}$
Quadratic 1D, $L = 64$	1.71 ± 0.23	1.27 ± 0.06	-1.56	$\mathbf{2.11}$
Quadratic 3D, $L = 128$	$\mathbf{-1.79 \pm 0.07}$	-2.14 ± 0.05	-472.76	-173.78
Furuta Dynamics	$\mathbf{8.25 \pm 0.33}$	7.55 ± 0.24	-6.16	-2.47

A.5 Experimental Details

We provide details about the data sets as well as about the experimental setup used in our experiments in Sec. 3.5.

A.5.1 Data Generation

In our experiments, we use several classes of functions to evaluate the architectures under consideration. To generate training data from these function classes, we sample L random tasks (as described in Sec. 3.5), and N_{tot} random input locations x for each task. For each minibatch of training tasks, we uniformly sample a context set size $N \in \{n_{\min}, \dots, n_{\max}\}$ and use a random subset of N data points from each task as context sets $\mathcal{D}^c$. The remaining $M = N_{\text{tot}} - N$ data points are used as the target sets $\mathcal{D}^t$ (cf. App. A.5.3 for the special case of the VI likelihood approximation). Tab. A.2 provides details about the data generation process.

A.5.1.1 GP Samples

We sample one-dimensional functions $f : \mathbb{R} \to \mathbb{R}$ from GP priors with three different stationary kernel functions as proposed by Gordon et al. [Gor20].

A radial basis functions (RBF) kernel with lengthscale $l = 1.0$:

$$k_{\text{RBF}}(r) \equiv \exp\left(-0.5r^2\right). \tag{A.11}$$

A weakly periodic kernel:

$$k_{\text{WP}}(r) \equiv \exp\left(-2\sin(0.5r)^2 - 0.125r^2\right). \tag{A.12}$$

A Matern-5/2 kernel with lengthscale $l = 0.25$:

$$k_{\text{M5/2}}(r) \equiv \left(1 + \frac{\sqrt{5}r}{0.25} + \frac{5r^2}{3 \cdot 0.25^2}\right)\exp\left(-\frac{\sqrt{5}r}{0.25}\right). \tag{A.13}$$

A.5.1.2 Quadratic Functions

We consider two classes of quadratic functions. The first class $f^{\text{Q,1D}} : \mathbb{R} \to \mathbb{R}$ is defined on a one-dimensional domain and parameterized by three parameters $a, b, c \in \mathbb{R}$:

$$f^{\text{Q,1D}}(x) \equiv a^2(x + b)^2 + c. \tag{A.14}$$

The second class $f^{Q,3D} : \mathbb{R}^3 \rightarrow \mathbb{R}$ is defined on a three-dimensional domain and also parameterized by three parameters $a, b, c \in \mathbb{R}$:

$$f^{Q,3D}(x_1, x_2, x_3) \equiv 0.5a\left(x_1^2 + x_2^2 + x_3^2\right) + b\left(x_1 + x_2 + x_3\right) + 3c. \tag{A.15}$$

This function class was proposed in Perrone et al. [Per18].

For both function classes we add Gaussian noise with standard deviation σ_n to the evaluations, cf. Tab. A.2.

Table A.2: Input spaces and parameters used to generate data for training and testing the architectures discussed in the main part of this paper. $\mathrm{U}(a, b)$ denotes the uniform distribution on the interval $[a, b]$, and, likewise $\mathrm{U}\{a, a+n\}$ denotes the uniform distribution on the set $\{a, a+1, \dots, a+n\}$.

Symbol	**Description**	**Value/Sampling distribution**
	GP Samples	
x	Input	$\mathrm{U}(-2.0, +2.0)$
N_{tot}	Number of data points per task	128
$\{n_{\min}, \dots n_{\max}\}$	Context set sizes during training	$\{3, \dots, 50\}$
	1D Quadratic Functions	
x	Input	$\mathrm{U}(-1.0, +1.0)$
a	Parameter	$\mathrm{U}(-0.5, +1.5)$
b	Parameter	$\mathrm{U}(-0.9, +0.9)$
c	Parameter	$\mathrm{U}(-1.0, +1.0)$
σ_n	Noise standard deviation	0.01
N_{tot}	Number of data points per task	128
$\{n_{\min}, \dots n_{\max}\}$	Context set sizes during training	$\mathrm{U}\{0, \dots, 20\}$
	3D Quadratic Functions	
x_1, x_2, x_3	Inputs	$\mathrm{U}(-1.0, +1.0)$
a, b, c	Parameters	$\mathrm{U}(+0.1, +10.0)$
σ_n	Noise standard deviation	0.01
N_{tot}	Number of data points per task	128
$\{n_{\min}, \dots n_{\max}\}$	Context set sizes during training	$\mathrm{U}\{0, \dots, 20\}$
	Furuta Dynamics	
$\theta_{\mathrm{arm}}, \theta_{\mathrm{pend}}$	Input angles	$\mathrm{U}(0.0, 2\pi \text{ rad})$
$\dot{\theta}_{\mathrm{arm}}, \dot{\theta}_{\mathrm{pend}}$	Input angular velocities	$\mathrm{U}(-2\pi \text{ rad}/0.5\text{ s}, 2\pi \text{ rad}/0.5\text{ s})$
m_{arm}	Mass arm	$\mathrm{U}\left(6.0 \cdot 10^{-2}\text{ kg}, 6.0 \cdot 10^{-1}\text{ kg}\right)$
m_{pend}	Mass pendulum	$\mathrm{U}\left(1.5 \cdot 10^{-2}\text{ kg}, 1.5 \cdot 10^{-1}\text{ kg}\right)$
l_{arm}	Length arm	$\mathrm{U}\left(5.6 \cdot 10^{-2}\text{ m}, 5.6 \cdot 10^{-1}\text{ m}\right)$
L_{arm}	Distance joint arm — mass arm	$\mathrm{U}\left(1.0 \cdot 10^{-1}\text{ m}, 3.0 \cdot 10^{-1}\text{ m}\right)$
L_{pend}	Distance joint pend.— mass pend.	$\mathrm{U}\left(1.0 \cdot 10^{-1}\text{ m}, 3.0 \cdot 10^{-1}\text{ m}\right)$
b_{arm}	Damping constant arm	$\mathrm{U}\left(2.0 \cdot 10^{-5}\text{ Nms}, 2.0 \cdot 10^{-3}\text{ Nms}\right)$
b_{pend}	Damping constant pendulum	$\mathrm{U}\left(5.6 \cdot 10^{-5}\text{ Nms}, 5.6 \cdot 10^{-3}\text{ Nms}\right)$
$\sigma_{\tau,\mathrm{arm}}$	Action noise standard dev.arm	0.5 Nm
$\sigma_{\tau,\mathrm{pend}}$	Action noise standard dev.pend.	0.5 Nm
N_{tot}	Number of data points per task	256
$\{n_{\min}, \dots n_{\max}\}$	Context set sizes during training	$\mathrm{U}\{0, \dots, 20\}$
	2D Image Completion MNIST	
x_1, x_2	Input pixel locations	$\mathrm{U}\{0, 27\}$ (scaled to $[0, 1]$)
N_{tot}	Number of data points per task	$28 \cdot 28$
$\{n_{\min}, \dots n_{\max}\}$	Context set sizes during training	$\mathrm{U}\{0, \dots, 28 \cdot 28/2\}$

A.5.1.3 Furuta Pendulum Dynamics

We consider a function class obtained by integrating the non-linear equations of motion governing the dynamics of a Furuta pendulum [Fur92, Caz11] for a time span of $\Delta t = 0.1$ s. More concretely, we consider the mapping

$$\Theta(t) \rightarrow \Theta(t + \Delta t) - \Theta(t), \tag{A.16}$$

where

$$\Theta = \left[\theta_{\text{arm}}(t), \theta_{\text{pend}}(t), \dot{\theta}_{\text{arm}}(t), \dot{\theta}_{\text{pend}}(t)\right]^T \tag{A.17}$$

denotes the four-dimensional vector describing the dynamical state of the Furuta pendulum. The Furuta pendulum is parameterized by seven parameters (two masses, three lengths, two damping constants) as detailed in Tab. A.2. During training, we provide $L = 64$ tasks, corresponding to 64 different parameter configurations. We consider the free system and generate noise by applying random torques at each integration time step ($\Delta t_{\text{Euler}} = 0.001$ s) to the joints of the arm and pendulum drawn from Gaussian distributions with standard deviations $\sigma_{\tau,\text{pend}}$, $\sigma_{\tau,\text{arm}}$, respectively.

A.5.1.4 2D Image Completion

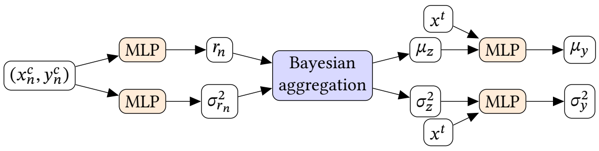
For this task, we use the MNIST database of 28×28 images of handwritten digits [LeC10], and define 2D functions mapping pixel locations $x_1, x_2 \in \{0, \ldots 27\}$ (scaled to the unit square) to the corresponding pixel intensities $y \in \{0, \ldots, 255\}$ (scaled to the unit interval), cf. Tab. A.2. One training task corresponds to one image drawn randomly from the training set (consisting of 60000 images) and for evaluation we use a subset of the test set (consisting of 10000 images).

A.5.2 Model Architectures

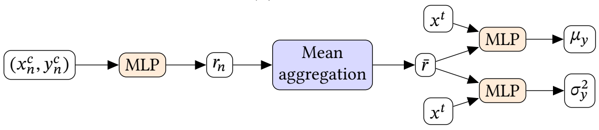
We provide the detailed architectures used for the experiments in Sec. 3.5 in Fig. A.1. For ANP we use multihead cross attention and refer the reader to Kim et al. [Kim19] for details about the architecture.

A.5.3 Hyperparameters and Hyperparameter Optimization

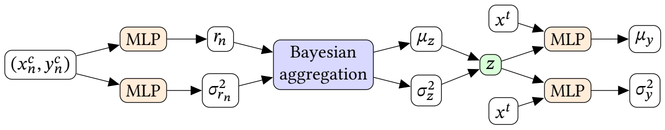
To arrive at a fair comparison of our BA with MA, it is imperative to use optimal model architectures for each aggregation method and likelihood approximation under consideration. Therefore, we optimize the number of hidden layers and the number of hidden units per layer of each encoder and decoder MLP (as shown in Fig. A.1), individually for each model architecture and each experiment. For the ANP, we also optimize the multihead attention MLPs. We further optimize the latent space dimensionality d_z and the learning rate of the Adam optimizer. For this hyperparameter optimization, we use the Optuna framework [Aki19] with TPE Sampler and Hyperband pruner [Li17b]. We consistently use a minibatch size of 16. Further, we use $S = 10$ latent samples to evaluate the MC likelihood approximation during training. To evaluate the VI likelihood approximation, we sample target set sizes between N_{tot} and N in each training epoch, cf. Tab. A.2.



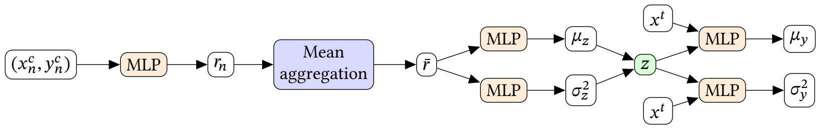
(a) BA + PB.



(b) MA + det. (CNP).



(c) BA + VI, BA + MC.



(d) MA + VI (LP-NP), MA + MC.

Figure A.1: Model architectures used for our experiments in Sec. 3.5. For the ANP architecture we refer the reader to Kim et al. [Kim19]. Orange rectangles denote MLPs. Blue rectangles denote aggregation operations. Variables in green rectangles are sampled from normal distributions with parameters given by the incoming nodes. To arrive at a fair comparison, we optimize all MLP architectures, the latent space dimensionality d_z, as well as the Adam learning rate, individually for all model architectures and all experiments, cf. App. A.5.3.

A.5.4 Evaluation Procedure

To evaluate the performance of the various model architectures we generate $L = 256$ unseen test tasks with target sets $\mathcal{D}_\ell^t$ consisting of $M = 256$ data points each and compute the average posterior predictive log-likelihood $\frac{1}{L}\frac{1}{M}\sum_{\ell=1}^{L} \log p\left(y_{\ell,1:M}^t \mid x_{\ell,1:M}^t, \mathcal{D}_\ell^c, \theta\right)$, given context sets $\mathcal{D}_\ell^c$ of size N.

Depending on the architecture, we approximate the posterior predictive log-likelihood according to:

- For BA + PB likelihood approximation:

$$\frac{1}{L}\frac{1}{M}\sum_{\ell=1}^{L}\sum_{m=1}^{M} \log p\left(y_{\ell,m}^t \mid x_{\ell,m}^t, \mu_{z,\ell}, \sigma_{z,\ell}^2, \theta\right). \tag{A.18}$$

- For MA + deterministic loss (= CNP):

$$\frac{1}{L}\frac{1}{M}\sum_{\ell=1}^{L}\sum_{m=1}^{M}\log p\left(y_{\ell,m}^{t} \mid x_{\ell,m}^{t}, \bar{r}_{\ell}, \theta\right). \tag{A.19}$$

- For architectures employing sampling-based likelihood approximations (VI, MC-LL) we report the joint log-likelihood over all data points in a test set, i.e.

$$\begin{aligned} &\frac{1}{L}\frac{1}{M}\sum_{\ell=1}^{L}\log\int q_{\phi}\left(z_{\ell} \mid \mathcal{D}_{\ell}^{c}\right)\prod_{m=1}^{M} p\left(y_{\ell,m}^{t} \mid x_{\ell,m}^{t}, z_{\ell}, \theta\right)\mathrm{d}z_{\ell} \\ &\approx \frac{1}{L}\frac{1}{M}\sum_{\ell=1}^{L}\log\frac{1}{S}\sum_{s=1}^{S}\prod_{m=1}^{M} p\left(y_{\ell,m}^{t} \mid x_{\ell,m}^{t}, z_{\ell,s}, \theta\right) \\ &= -\frac{1}{M}\log S + \frac{1}{L}\frac{1}{M}\sum_{l=1}^{L}\underset{s=1}{\overset{S}{\operatorname{logsumexp}}}\left(\sum_{m=1}^{M}\log p\left(y_{\ell,m}^{t} \mid x_{\ell,m}^{t}, z_{\ell,s}, \theta\right)\right), \end{aligned} \tag{A.20}$$

  where $z_{\ell,s} \sim q_{\phi}\left(z \mid \mathcal{D}_{\ell}\right)$. We employ $S = 25$ latent samples.

To compute the log-likelihood values given in tables, we additionally average over various context set sizes N as detailed in the main part of this paper.

We report the mean posterior predictive log-likelihood computed in this way w.r.t. 10 training runs with different random seeds together with 95% confidence intervals

A.6 Additional Experimental Results

We provide additional experimental results accompanying the experiments presented in Sec. 3.5:

- Results for relative evaluation runtimes and numbers of parameters of the optimized network architectures on the full GP suite of experiments, cf. Tab. A.3.
- The posterior predictive mean squared error on all experiments, cf. Tab. A.4.
- The context-size dependent results for the predictive posterior log-likelihood for the 1D and 3D Quadratic experiments, the Furuta dynamics experiment, as well as the 2D image completion experiment, cf. Fig. A.2.
- More detailed plots of the predictions on one-dimensional experiments (1D Quadratics (Figs. A.3, A.4), RBF-GP, (Figs. A.5, A.6), Weakly Periodic GP (Figs. A.7, A.8), and Matern-5/2 GP (Figs. A.9, A.10)).

Table A.3: Relative evaluation runtimes and numbers of parameters of the optimized network architectures on the GP tasks. The deterministic methods (PB, det.) are much more efficient regarding evaluation runtime, as they require only on forward pass per prediction, while the sampling-based approaches (VI, MC) require multiple forward passes (each corresponding to one latent sample) to compute their predictions. We use $S = 25$ latent samples, as described in App. A.5.4. Furthermore, BA tends to require less complex encoder and decoder network architectures compared to MA, because it represents a more efficient mechanism to propagate information from the context set to the latent state.

			BA	MA
RBF GP	Runtime	PB/det.	1	1.4
		VI	18	25
		MC	32	27
	#Parameters	PB/det.	72k	96k
		VI	63k	77k
		MC	122k	153k
Weakly Periodic GP	Runtime	PB/det.	1	1.4
		VI	11	10
		MC	22	15
	#Parameters	PB/det.	51k	87k
		VI	48k	72k
		MC	87k	89k
Matern-5/2 GP	Runtime	PB/det.	1	1.1
		VI	6.5	11
		MC	15	19
	#Parameters	PB/det.	53k	100k
		VI	32k	35k
		MC	108k	104k

Table A.4: Posterior predictive mean squared error (MSE) on all experiments presented in this paper. We average over the same context set sizes as used to compute the posterior predictive log-likelihood, cf. Sec. 3.5, and again use $S = 25$ latent samples to compute the mean prediction of sampling-based methods. Our BA consistently improves predictive performance compared to MA not only in terms of likelihood (as shown in Sec. 3.5), but also in terms of MSE. Furthermore, while ANP tends to perform poorly in terms of likelihood (cf. Sec. 3.5), it's MSE is improved greatly by the attention mechanism.

		BA	MA
RBF GP	PB/det.	**0.0623 ± 0.0009**	0.0687 ± 0.0010
	VI	**0.0736 ± 0.0005**	0.0938 ± 0.0036
	MC	**0.0637 ± 0.0007**	0.0741 ± 0.0012
	ANP		**0.0550 ± 0.0009**
Weakly Periodic GP	PB/det.	**0.0679 ± 0.0007**	0.0761 ± 0.0014
	VI	**0.0879 ± 0.0017**	**0.1326 ± 0.0518**
	MC	**0.0677 ± 0.0008**	0.0832 ± 0.0009
	ANP		**0.0592 ± 0.0009**
Matern-5/2 GP	PB/det.	**0.2452 ± 0.0088**	0.3021 ± 0.0035
	VI	**0.3702 ± 0.0100**	0.6292 ± 0.1077
	MC	**0.2321 ± 0.0019**	0.5166 ± 0.1438
	ANP		**0.1890 ± 0.0012**
Quadratics 1D, $L = 64$	PB/det.	**0.1447 ± 0.0095**	**0.1513 ± 0.0091**
	VI	**0.1757 ± 0.0128**	**0.1833 ± 0.0154**
	MC	**0.1473 ± 0.0107**	**0.1636 ± 0.0082**
	ANP		**0.1330 ± 0.0037**
Quadratics 3D, $L = 128$	PB/det.	**190.5 ± 1.4**	195.4 ± 1.5
	VI	**253.1 ± 18.0**	**278.1 ± 40.5**
	MC	**196.8 ± 2.6**	206.7 ± 5.3
	ANP		**192.5 ± 2.7**
Furuta Dynamics	PB/det.	**0.1742 ± 0.0092**	0.1989 ± 0.0095
	VI	**0.2269 ± 0.0088**	0.2606 ± 0.0165
	MC	**0.1758 ± 0.0124**	**0.1977 ± 0.0154**
	ANP		**0.1516 ± 0.0073**
2D Image Completion	PB/det.	**0.0348 ± 0.0010**	0.0417 ± 0.0026
	ANP		**0.0215 ± 0.0003**

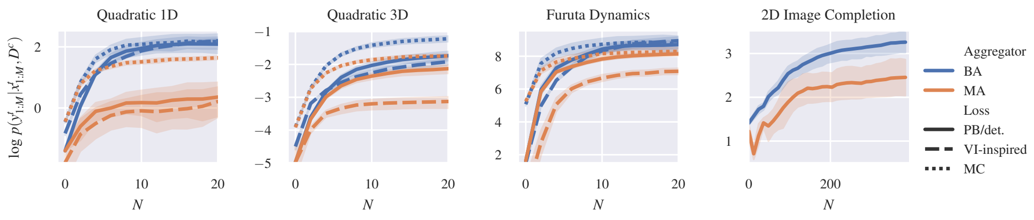
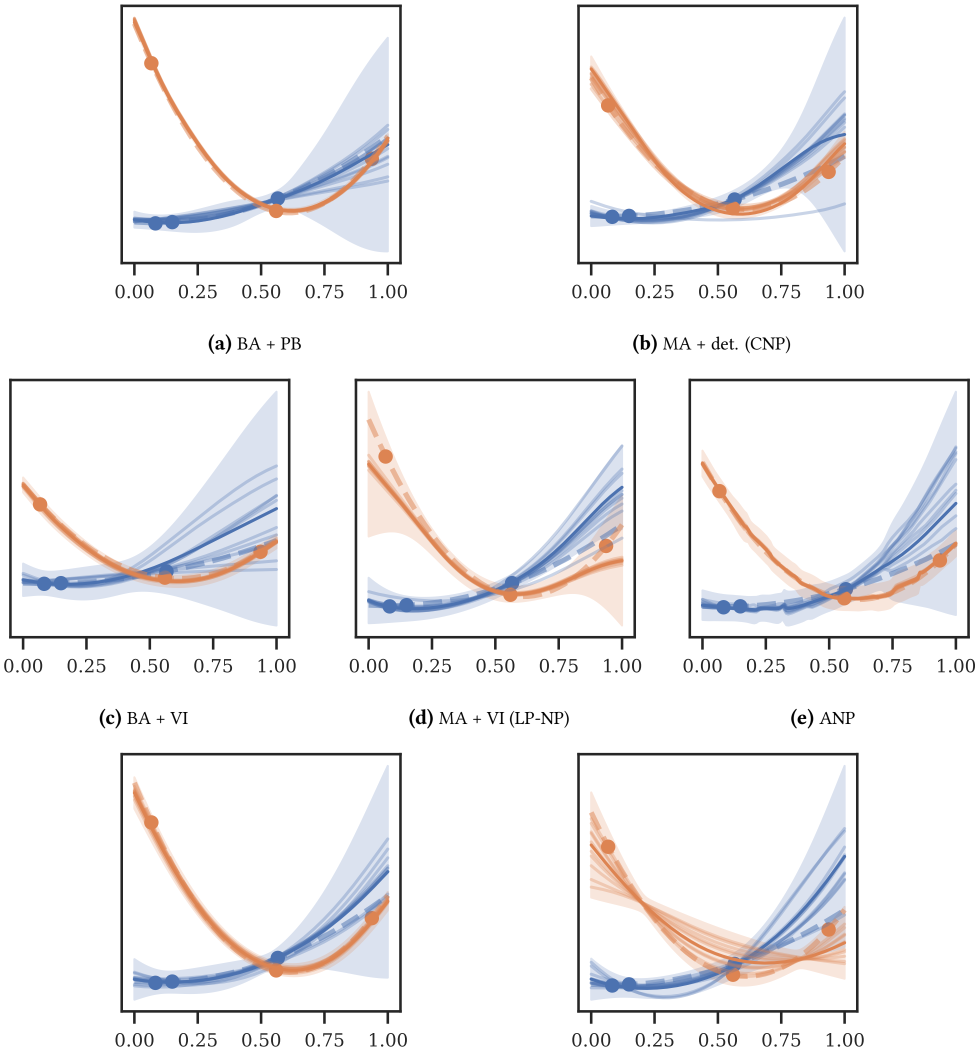


Figure A.2: Posterior predictive log-likelihood in dependence of the context set size N for the 1D and 3D Quadratic experiments, the Furuta dynamics experiment as well as the 2D image completion experiment.



(a) BA + PB

(b) MA + det. (CNP)

(c) BA + VI

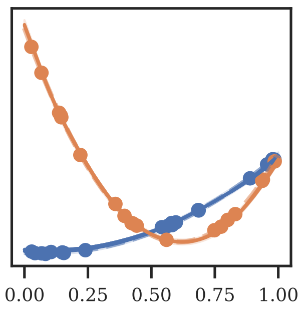
(d) MA + VI (LP-NP)

(e) ANP

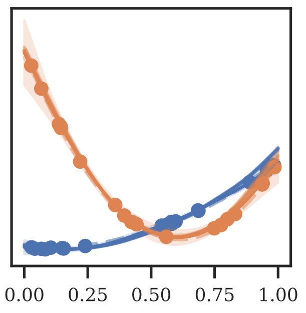
(f) BA + MC-LL

(g) MA + MC-LL

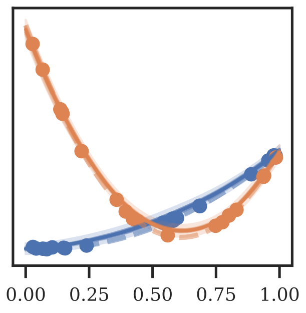
Figure A.3: Predictions on two instances (dashed lines) of the 1D quadratic function class, given $N = 3$ context data points (circles). We plot mean and standard deviation (solid line, shaded area) predictions together with 10 function samples (for deterministic methods we employ AR sampling).



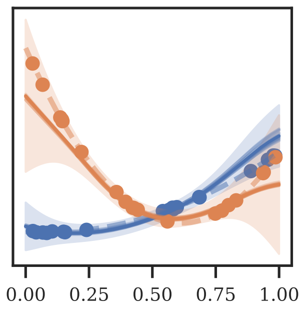
(a) BA + PB



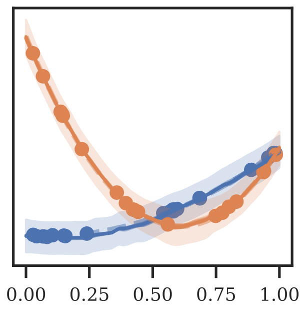
(b) MA + det. (CNP)



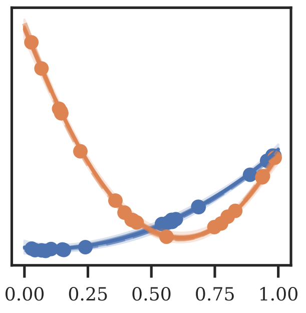
(c) BA + VI



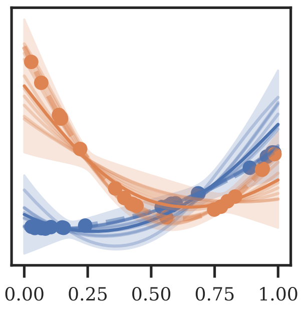
(d) MA + VI (LP-NP)



(e) ANP

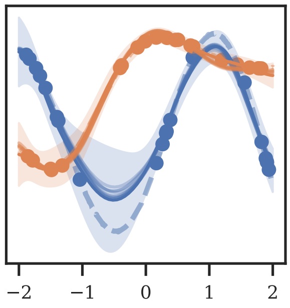


(f) BA + MC-LL

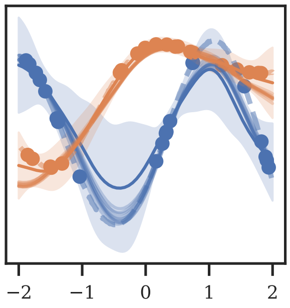


(g) MA + MC-LL

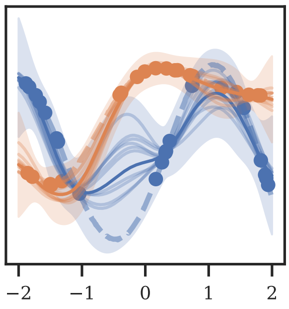
Figure A.4: Predictions on two instances (dashed lines) of the 1D quadratic function class, given $N = 19$ context data points (circles). We plot mean and standard deviation (solid line, shaded area) predictions together with 10 function samples (for deterministic methods we employ AR sampling).



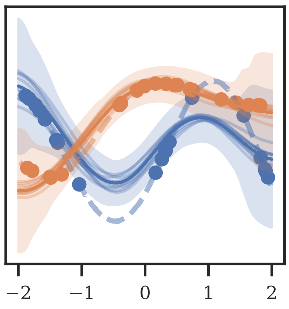
(a) BA + PB



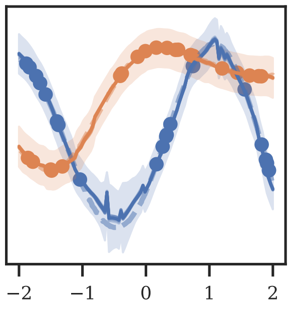
(b) MA + det. (CNP)



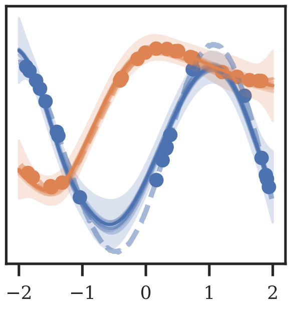
(c) BA + VI



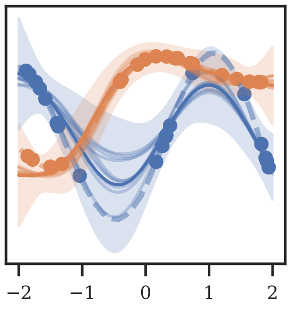
(d) MA + VI (LP-NP)



(e) ANP

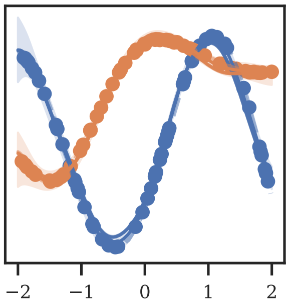


(f) BA + MC-LL

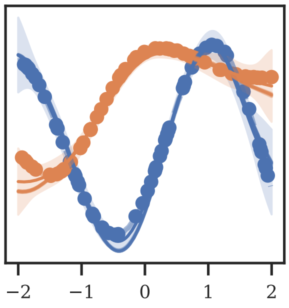


(g) MA + MC-LL

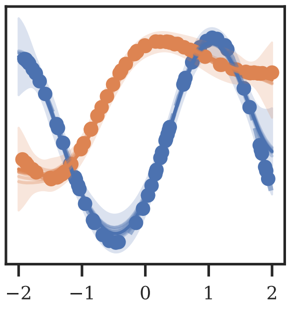
Figure A.5: Predictions on two instances (dashed lines) of the RBF GP function class, given $N = 20$ context data points (circles). We plot mean and standard deviation (solid line, shaded area) predictions together with 10 function samples (for deterministic methods we employ AR sampling).



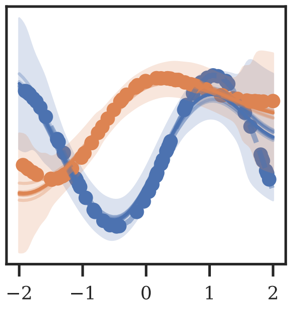
(a) BA + PB



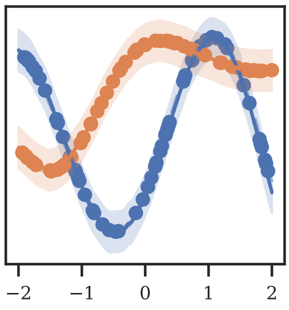
(b) MA + det. (CNP)



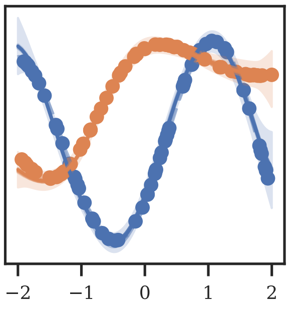
(c) BA + VI



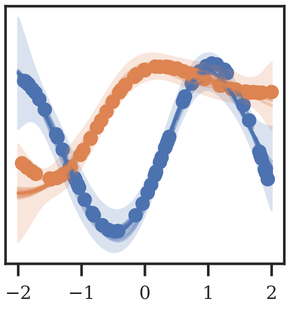
(d) MA + VI (LP-NP)



(e) ANP

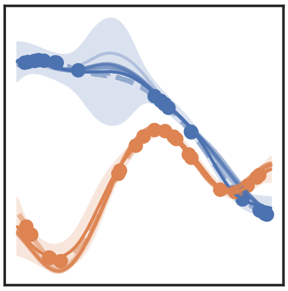


(f) BA + MC-LL

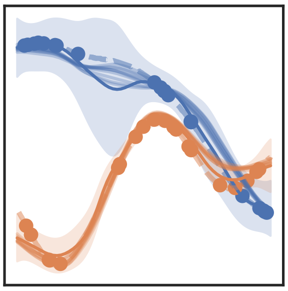


(g) MA + MC-LL

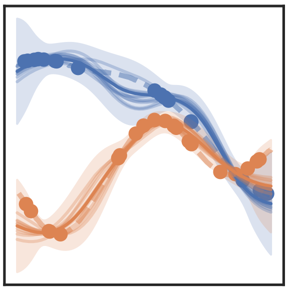
Figure A.6: Predictions on two instances (dashed lines) of the RBF GP function class, given $N = 60$ context data points (circles). We plot mean and standard deviation (solid line, shaded area) predictions together with 10 function samples (for deterministic methods we employ AR sampling).



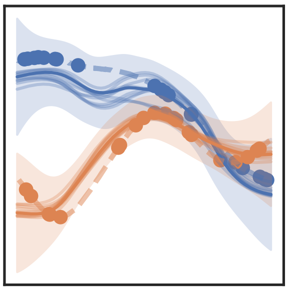
(a) BA + PB



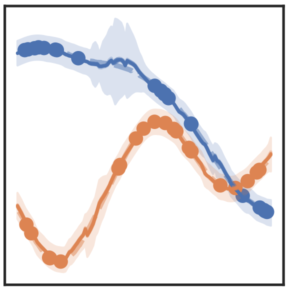
(b) MA + det. (CNP)



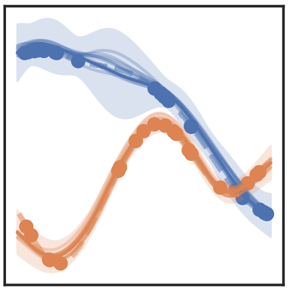
(c) BA + VI



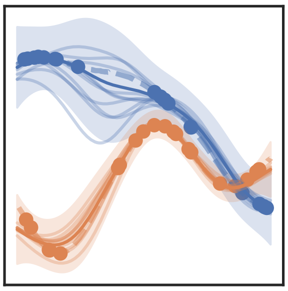
(d) MA + VI (LP-NP)



(e) ANP

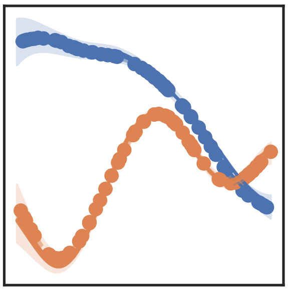


(f) BA + MC-LL

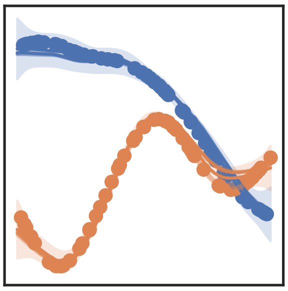


(g) MA + MC-LL

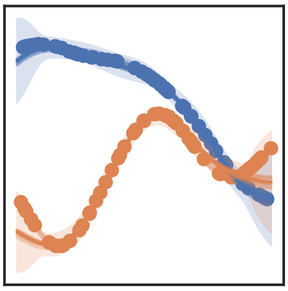
Figure A.7: Predictions on two instances (dashed lines) of the Weakly Periodic GP function class, given $N = 20$ context data points (circles). We plot mean and standard deviation (solid line, shaded area) predictions together with 10 function samples (for deterministic methods we employ AR sampling).



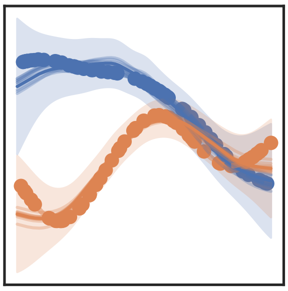
(a) BA + PB



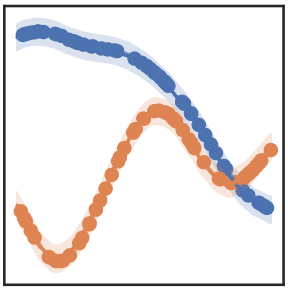
(b) MA + det. (CNP)



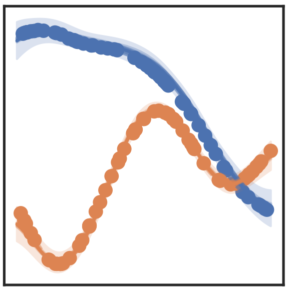
(c) BA + VI



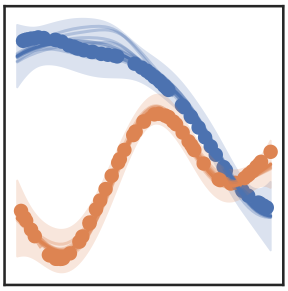
(d) MA + VI (LP-NP)



(e) ANP

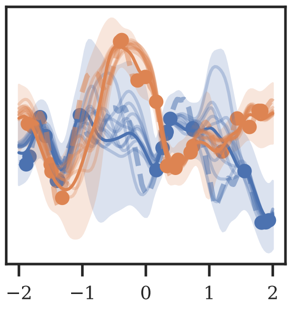


(f) BA + MC-LL

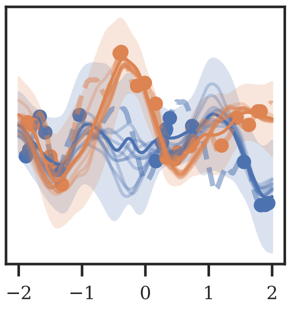


(g) MA + MC-LL

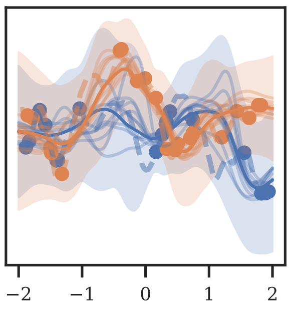
Figure A.8: Predictions on two instances (dashed lines) of the Weakly Periodic GP function class, given $N = 60$ context data points (circles). We plot mean and standard deviation (solid line, shaded area) predictions together with 10 function samples (for deterministic methods we employ AR sampling).



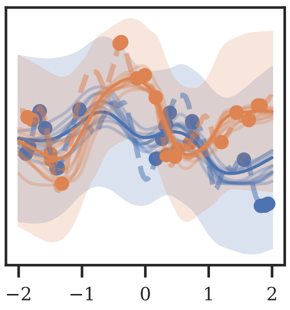
(a) BA + PB



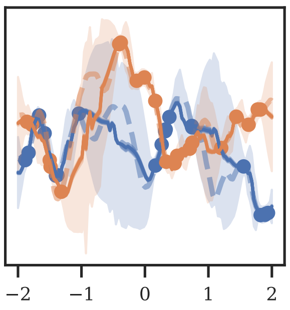
(b) MA + det. (CNP)



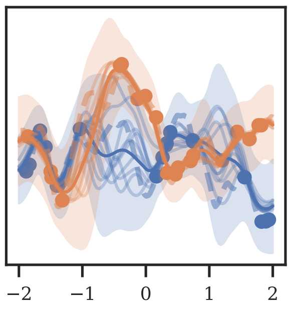
(c) BA + VI



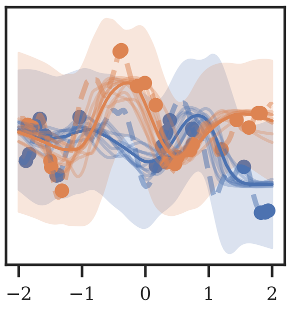
(d) MA + VI (LP-NP)



(e) ANP

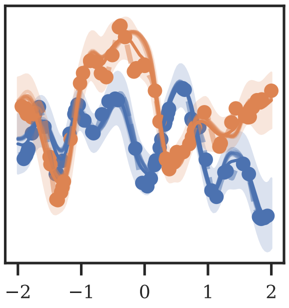


(f) BA + MC-LL

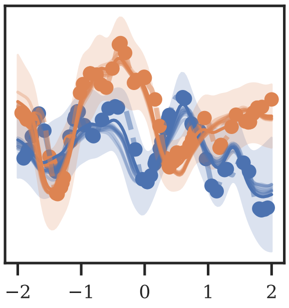


(g) MA + MC-LL

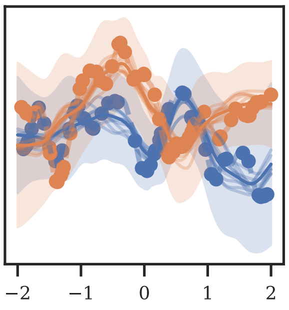
Figure A.9: Predictions on two instances (dashed lines) of the Matern-5/2 GP function class, given $N = 20$ context data points (circles). We plot mean and standard deviation (solid line, shaded area) predictions together with 10 function samples (for deterministic methods we employ AR sampling).



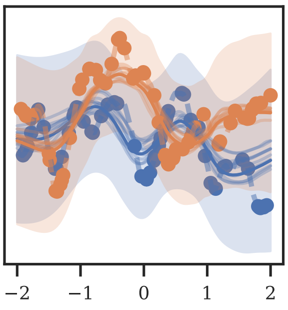
(a) BA + PB



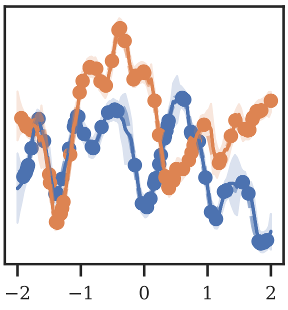
(b) MA + det. (CNP)



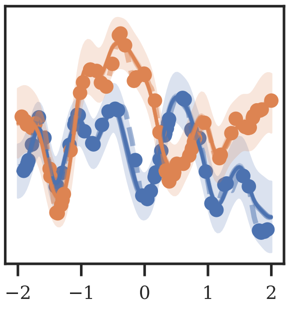
(c) BA + VI



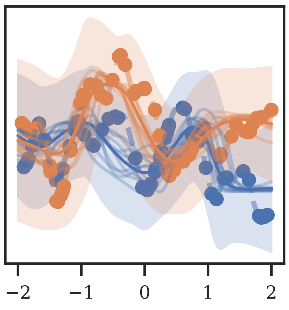
(d) MA + VI (LP-NP)



(e) ANP



(f) BA + MC-LL



(g) MA + MC-LL

Figure A.10: Predictions on two instances (dashed lines) of the Matern-5/2 GP function class, given $N = 60$ context data points (circles). We plot mean and standard deviation (solid line, shaded area) predictions together with 10 function samples (for deterministic methods we employ AR sampling).

B Accurate Task Posterior Inference with Gaussian Mixture Models

This appendix provides further details that supplement the main part of our paper.

B.1 Algorithmic Details

In this section we lay out the full set of variational update equations and provide pseudocode for our GMM-NP algorithm.

B.1.1 Variational Update Equations

We provide the full set of equations required to compute the TRNG update for the variational parameters $\boldsymbol{\phi}_\ell \equiv \{w_{\ell,k}, \boldsymbol{\mu}_{\ell,k}, \boldsymbol{\Sigma}_{\ell,k}\}$, $k \in \{1, \dots, K\}$, parameterizing our GMM TP approximation as

$$q_{\boldsymbol{\phi}_\ell}(\boldsymbol{z}_\ell) \equiv \sum_k w_{\ell,k} q_{\boldsymbol{\phi}_\ell}(\boldsymbol{z}_\ell \mid k) \equiv \sum_k w_{\ell,k} \mathcal{N}(\boldsymbol{z}_\ell \mid \boldsymbol{\mu}_{\ell,k}, \boldsymbol{\Sigma}_{\ell,k}) \tag{B.1}$$

with $\sum_k w_{\ell,k} = 1$. The TRNG-VI update equations, as proposed by Arenz et al. [Are23], read

$$\boldsymbol{\Sigma}_{\ell,k,\text{new}} = \left[\frac{\eta}{\eta+1}\boldsymbol{\Sigma}^{-1}_{\ell,k,\text{old}} - \frac{1}{\eta+1}\boldsymbol{R}_{\ell,k}\right]^{-1}, \tag{B.2a}$$

$$\boldsymbol{\mu}_{\ell,k,\text{new}} = \boldsymbol{\Sigma}_{\ell,k,\text{new}}\left[\frac{\eta}{\eta+1}\boldsymbol{\Sigma}^{-1}_{\ell,k,\text{old}}\boldsymbol{\mu}_{\ell,k,\text{old}} + \frac{1}{\eta+1}\left(\boldsymbol{r}_{\ell,k} - \boldsymbol{R}_{\ell,k}\boldsymbol{\mu}_{\ell,k,\text{old}}\right)\right], \tag{B.2b}$$

$$w_{\ell,k,\text{new}} \propto \exp \rho_{\ell,k}, \tag{B.2c}$$

where $\boldsymbol{R}_{\ell,k}$, $\boldsymbol{r}_{\ell,k}$, and $\rho_{\ell,k}$ are defined as expectations that can be approximated from per-component samples using MC:

$$\boldsymbol{R}_{\ell,k} = \mathbb{E}_{q_{\boldsymbol{\phi}_{\ell,\text{old}}}(\boldsymbol{z}_\ell \mid k)}\left[\boldsymbol{\Sigma}^{-1}_{\ell,k,\text{old}}(\boldsymbol{z}_\ell - \boldsymbol{\mu}_{\ell,k,\text{old}})\nabla^T_{\boldsymbol{z}_\ell} h_{\ell,k}(\boldsymbol{z}_\ell)\right], \tag{B.3a}$$

$$\boldsymbol{r}_{\ell,k} = \mathbb{E}_{q_{\boldsymbol{\phi}_{\ell,\text{old}}}(\boldsymbol{z}_\ell \mid k)}\left[\nabla_{\boldsymbol{z}_\ell} h_{\ell,k}(\boldsymbol{z}_\ell)\right], \tag{B.3b}$$

$$\rho_{\ell,k} = \mathbb{E}_{q_{\boldsymbol{\phi}_{\ell,\text{old}}}(\boldsymbol{z}_\ell \mid k)}\left[h_{\ell,k}(\boldsymbol{z}_\ell) - \log q_{\boldsymbol{\phi}_{\ell,\text{old}}}(\boldsymbol{z}_\ell \mid k)\right]. \tag{B.3c}$$

Here, we defined

$$h_{\ell,k}(\boldsymbol{z}_\ell) \equiv \log \tilde{p}_\ell(\boldsymbol{z}_\ell) + \log q_{\boldsymbol{\phi}_{\ell,\text{old}}}(\boldsymbol{z}_\ell|k) - \log q_{\boldsymbol{\phi}_{\ell,\text{old}}}(\boldsymbol{z}_\ell)\,. \tag{B.4}$$

The optimal value for the Lagrangean parameter $\eta \geq 0$ that enforces the trust region constraint

$$\text{KL}\left[q_{\boldsymbol{\phi}} \| q_{\boldsymbol{\phi}_\text{old}}\right] \leq \varepsilon, \tag{B.5}$$

is defined by a scalar convex optimization problem that can be solved efficiently by a bracketing search, which also ensures positive definiteness of the new covariance matrix $\boldsymbol{\Sigma}_{\ell,k,\text{new}}$.

B.1.2 GMM Initialization

We provide details on the initialization of the variational GMMs before meta-training and testing. As we use the same procedure for each task, we drop task indices ℓ to avoid clutter. Given a number K of components for the GMM task posterior (TP) $q_{\boldsymbol{\phi}}(z)$ Eq. (4.12), we use a prior $p(z)$ with K components. To initialize the means μ_k, covariances Σ_k, and mixture weights w_k for $k \in \{1, \dots, K\}$, we use the same simple heuristic as Arenz et al. [Are23]:

- Draw the means μ_k from a d_z-dimensional standard Normal distribution,
- The covariances Σ_k are initialized as diagonal matrices (with 1 on the diagonal),
- The weights are initialized uniformly as $w_k = 1/K$.

B.1.3 Algorithm Summary

We provide pseudocode for the meta-training stage of our GMM-NP algorithm in Alg. 1 and for the prediction stage in Alg. 2.

B.1.4 Discussion of Convergence Properties

B.1.4.1 Convergence of the ELBO

Our algorithm inherits the convergence guarantee of the variational Bayes algorithm as discussed, e.g., in Bishop [Bis06]. In general, convergence of variational Bayes is independent of the concrete optimization strategy used for (ϕ, θ): as long as both the E-step (step in ϕ) and the M-step (step in θ) increase the ELBO objective (first term in Eq. (4.14)), the algorithm is guaranteed to converge to a local optimum of the ELBO. While in standard, reparameterized, variational Bayes (as employed by the baseline methods studied in Sec. 4.5) (ϕ, θ) are optimized jointly using, e.g., Adam [Kin15], our method alternates between a step in ϕ using TRNG-VI [Are23] and a step in θ using Adam. Nevertheless, both steps increase the ELBO, so our algorithm will converge.

Algorithm 1 GMM-NP (Meta-Training)

```
Require: Meta-data $\mathcal{D}_\ell = \{\boldsymbol{x}_{\ell,1:N}, \boldsymbol{y}_{\ell,1:N}\}$, $\ell \in 1:L$
  Sample variably-sized auxiliary tasks $\tilde{\mathcal{D}}_{\tilde{\ell}} = \{\boldsymbol{x}_{\tilde{\ell},1:N_{\tilde{\ell}}}, \boldsymbol{y}_{\tilde{\ell},1:N_{\tilde{\ell}}}\}$, $\tilde{\ell} \in 1:\tilde{L}$, cf. Sec. B.3.2
  Initialize variational parameters $\boldsymbol{\phi}_{1:\tilde{L}} = \{w_{1:\tilde{L},1:K}, \boldsymbol{\mu}_{1:\tilde{L},1:K}, \boldsymbol{\Sigma}_{1:\tilde{L},1:K}\}$
  Initialize model parameters $\boldsymbol{\theta}$
  while not converged do
      for each minibatch of tasks $I \subset \{1, \dots, \tilde{L}\}$ do
          Sample $\boldsymbol{z}_{\ell,k,s} \sim q_{\boldsymbol{\phi}_\ell}(\boldsymbol{z}_\ell \mid k)$ for $\ell \in I$, $k \in 1:K$, $s \in 1:S$
          Evaluate $h_{\ell,k}(\boldsymbol{z}_{\ell,k,s}, \tilde{\mathcal{D}}_\ell)$ for $\ell \in I$, $k \in 1:K$, $s \in 1:S$, Eq. (B.4)
          Update variational parameters $\boldsymbol{\phi}_\ell$ for $\ell \in I$, Eq. (4.13)
          Sample $\boldsymbol{z}_{\ell,s} \sim q_{\boldsymbol{\phi}_\ell}(\boldsymbol{z}_\ell)$ for $\ell \in I$, $s \in 1:S$
          Estimate $\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}), \propto \sum_{s,n} \nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\boldsymbol{y}_{\ell,n} \mid \boldsymbol{x}_{\ell,n}, \boldsymbol{z}_{\ell,s})$, Eq. (4.14)
          Perform step in $\boldsymbol{\theta}$ using Adam
      end for
  end while
  return Model parameters $\boldsymbol{\theta}$
```

Algorithm 2 GMM-NP (Prediction)

```
Require: Context data $\mathcal{D}_*^c = \{\boldsymbol{x}_{*,1:M_*}^c, \boldsymbol{y}_{*,1:M_*}^c\}$, model parameters $\boldsymbol{\theta}$, target inputs $\boldsymbol{x}_{*,1:N_*}^t$
  Initialize variational parameters $\boldsymbol{\phi}_* = \{w_{*,1:K}, \boldsymbol{\mu}_{*,1:K}, \boldsymbol{\Sigma}_{*,1:K}\}$
  while not converged do
      Sample $\boldsymbol{z}_{*,k,s} \sim q_{\boldsymbol{\phi}_*}(\boldsymbol{z}_* \mid k)$ for $k \in 1:K$, $s \in 1:S$
      Evaluate $h_{*,k}(\boldsymbol{z}_{*,k,s}, \mathcal{D}_*^c)$ for $k \in 1:K$, $s \in 1:S$, Eq. (B.4)
      Update variational parameters $\boldsymbol{\phi}_*$, Eq. (4.13)
  end while
  Sample $\boldsymbol{z}_* \sim q_{\boldsymbol{\phi}_*}(\boldsymbol{z}_*)$
  return Predictions $\boldsymbol{y}_{*,n}^t = \mathrm{dec}_{\boldsymbol{\theta}}(\boldsymbol{x}_{*,n}^t, \boldsymbol{z}_*)$, $n \in 1:N_*$
```

B.1.4.2 Convergence of the Marginal Likelihood

As discussed in Sec. 4.4, our GMM-NP algorithm is designed to improve the convergence behaviour w.r.t. the *marginal likelihood* Eq. (4.14) in comparison to existing NP-based BML approaches. Recall that the convergence guarantee of the classical expectation-maximization (EM) algorithm w.r.t. the marginal likelihood is lost as soon as the E-step becomes intractable, i.e., as soon as the posterior distribution cannot be computed exactly, and, thus, has to be approximated by a variational distribution, cf., e.g., [Bis06]. This is the case for most models of reasonable complexity, e.g., for the variational autoencoder [Kin13] or the NP model family [Gar18c]. Our GMM-NP model is no exception here, as we build on the NP model for which the TP distribution cannot be computed analytically. Convergence of the marginal likelihood when using the ELBO (first term in Eq. (4.14)) as a surrogate objective is guaranteed if the ELBO is tight after the E-step, which is the setting of the aforementioned EM algorithm and only the case for a perfect TP approximation, i.e., if $\mathrm{KL}\left[q_\phi(z) \| p_\theta(z \mid \mathcal{D}^c)\right] = 0$, cf. also App. B.3.3. For imperfect approximations, the tightness of the bound is controlled by the variational gap $\mathrm{KL}\left[q_\phi(z) \| p_\theta(z \mid \mathcal{D}^c)\right] > 0$. A better approximate posterior $q_\phi(z)$ yields a tighter ELBO, which in turn brings us closer to the EM setting, i.e., typically improves convergence. Our GMM-NP algorithm builds exactly on this insight: we use an expressive TP approximation by a full-covariance GMM and a powerful optimizer for ϕ (TRNG-VI, [Are23]) to obtain a tighter ELBO than existing BML approaches in order to achieve optimization of the model parameters in a way that efficiently maximizes the marginal likelihood.

B.2 Baseline Algorithms

Tab. B.1 gives an overview of the architectural differences of the BML approaches we compared in our empirical evaluation (Sec. 4.5).

Table B.1: Comparison of state-of-the-art approaches for Bayesian meta-learning (TRNGD = trust region natural gradient descent, RSGD = reparameterized stochastic gradient descent, SGD = stochastic gradient descent, SE = set encoder, MA = mean aggregation, BA = Bayesian aggregation, SA = self attention, CA = cross attention).

	TP Approx.	VI Approach	Amortization	Det. Path
GMM-NP (ours)	Full-cov. GMM	TRNGD	none	none
MA-NP [Gar18c]	Diag. Gaussian	RSGD	SE + MA	none
BA-NP [Vol21]	Diag. Gaussian	RSGD	SE + BA	none
BNP [Lee20]	Non-parametric	SGD	SE + MA	none
ANP [Kim19]	Diag. Gaussian	RSGD	SA + SE + MA	CA
BANP [Lee20]	Non-parametric	SGD	SA + SE + MA	CA

To compute our results, we consistently use code by the original authors. We also provide source code for our proposed GMM-NP algorithm:

- Source code four our GMM-NP algorithm:
  https://github.com/ALRhub/gmm_np,
- MA-NP, ANP:
  https://github.com/deepmind/neural-processes,
- BA-NP:
  https://github.com/boschresearch/bayesian-context-aggregation,
- BNP, BANP:
  https://github.com/juho-lee/bnp.

B.3 Experimental Protocol

To foster reproducibility, we provide details on our experimental protocol.

B.3.1 Model Hyperparameters

To arrive at a fair comparison of our GMM-NP model with the baseline approaches, we optimize model hyperparameters individually for each model-dataset combination presented in Sec. 4.5. Concretely, we perform a Bayesian hyperparameter sweep with 256 trials for each model-dataset combination over the parameters detailed below. For the image completion experiment on MNIST, we employ a grid search with fewer trials to keep the computational effort manageable. For hyperparameters not mentioned below, we consistently use standard settings proposed by the original authors. To implement the hyperparameter search, we use the wandb sweep functionality [Bie20].

B.3.1.1 Observation Noise Parametrization

As detailed in Sec. 4.3.2, all compared models (including our GMM-NP) employ a Gaussian likelihood of the form

$$p_\theta(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{z}) \equiv \mathcal{N}\left(\boldsymbol{y} \mid \mathrm{dec}_\theta^{\boldsymbol{\mu}}(\boldsymbol{x}, \boldsymbol{z}), \mathrm{diag}\left(\sigma_\mathrm{n}^2\right)\right), \tag{B.6}$$

where the mean is computed by a decoder DNN $\mathrm{dec}_\theta^{\boldsymbol{\mu}}$ receiving the input location $\boldsymbol{x}$ and a latent sample $\boldsymbol{z}$. However, different parameterizations of the observation noise variance σ_n^2 are used in the literature. As it is not clear which setting is fairest, we also treat the observation noise parameterization as a hyperparameter. Concretely, for each model-dataset combination, we test the following settings for the observation noise (with individual hyperparameter sweeps) and report the best performing one:

1 $\sigma_\mathrm{n}^2 = \sigma_\mathrm{n,true}^2$ with $\sigma_\mathrm{n,true}^2$ being the true noise variance on the data,

2 $\sigma_\mathrm{n}^2 \in \mathbb{R}$ is a single float value, optimized jointly with $\boldsymbol{\theta}$,

3 $\sigma_\mathrm{n}^2 = \mathrm{dec}_\theta^{\boldsymbol{\sigma}}(\boldsymbol{x})$, i.e., observation noise is parameterized by a second decoder network, optimized jointly with $\mathrm{dec}_\theta^{\boldsymbol{\mu}}$, but receiving only the input location,

4 $\sigma_\mathrm{n}^2 = \mathrm{dec}_\theta^{\boldsymbol{\sigma}}(\boldsymbol{x}, \boldsymbol{z})$, i.e., observation noise is parameterized by a second decoder network, optimized jointly with $\mathrm{dec}_\theta^{\boldsymbol{\mu}}$, and also receiving both the input location and the latent sample.

For all compared models, and regardless of the parameterization, we bound the observation noise from below using a softplus transformation s.t. $\sigma_\mathrm{n} \geq \sigma_\mathrm{n,min} = 0.1$, as proposed by [Gar18c, Kim19, Lee20].

B.3.1.2 DNN Architectures

For all experiments and all baseline models, we use encoder and decoder DNNs with two hidden layers. Likewise, our GMM-NP model uses a decoder DNN with two hidden layers. We optimize the number of hidden units per layer within the bounds $\{8, \dots, 64\}$.

B.3.1.3 Latent Dimensionalities

For baseline models with parametric latent distributions (all except B(A)NP), we optimize the latent dimension d_z within the bounds $\{1, \dots, 64\}$. As our GMM-NP algorithm employs full covariance matrices, we restrict the bounds for d_z to $\{1, \dots, 8\}$ for a fair comparison.

B.3.1.4 Number of GMM components

For our GMM-NP algorithm, as well as for iBayes-GMM [Lin20] used for the comparison in Sec. B.5.1, we optimize the number of GMM components within the bounds $\{1, \dots, 10\}$.

B.3.1.5 Learning Rates and Trust Region Bounds

All algorithms use the Adam optimizer with standard settings to update DNN weights. We optimize the corresponding learning rates on a log-uniform scale within the bounds $\left[10^{-5}, 10^{-1}\right]$. We use the same settings to optimize the step size for the GMM updates of the variational parameters of the iBayes-GMM algorithm [Lin20] used for the comparison in Sec. B.5.1. As proposed by Arenz et al. [Are23], we optimize the Lagrangean parameter η of our GMM-NP algorithm using a bracketing search on the interval $\left[10^{-3}, 10^{-1}\right]$.

B.3.2 Auxiliary Subtask Generation for Meta-Training

We describe the procedure to sample auxiliary subtasks during meta-training in more detail, cf. Sec. 4.4.

B.3.2.1 Nomenclature

Recall from Sec. 4.3.2 that we define a *meta-task* as the set of all available (noisy) evaluations $\mathcal{D}_\ell$, $\ell \in \{1, \dots, L\}$ from an unknown function f_ℓ and that each meta-task contains N examples. Thus, a meta-task $\mathcal{D}_\ell$ is all data a BML algorithm has available to learn about f_ℓ during meta-training. A *subtask* of meta-task $\mathcal{D}_\ell$ is defined as an arbitrary subset of $\mathcal{D}_\ell$.

B.3.2.2 Auxiliary Subtask Sampling

As described in Sec. 4.4, standard NP meta-training samples auxiliary subtasks from the meta-data for each minibatch step in order to provide the decoder with samples from task posterior approximations informed by a range of context sizes. We use the following standard procedure [Gar18c, Kim19, Lee20] to sample auxiliary subtasks to evaluate the optimization objectives of the baseline approaches (e.g., Eq. (4.8) for standard NP). Given a minibatch $I \subset \{1, \dots, L\}$ of meta-tasks $\mathcal{D}_\ell$, $\ell \in I$, we first sample auxiliary subtasks $\tilde{\mathcal{D}}_\ell$ with a size $\tilde{N}$ drawn uniformly from $\tilde{N} \in \{N_{\min} + 1, \dots, N_{\max}\}$ with $N_{\min} \geq 1$ and $N_{\max} \leq N$. Then, we sample context sets $\tilde{\mathcal{D}}_\ell^c \subset \tilde{\mathcal{D}}_\ell$ of size M, drawn uniformly from $M \in \left\{1, \dots, \tilde{N}\right\}$. $\tilde{\mathcal{D}}_\ell^c$ and $\tilde{\mathcal{D}}_\ell$ are then used in Eq. (4.8) to compute the ELBO objective for the current minibatch.

As described in Sec. 4.4, our GMM-NP algorithm uses a similar approach: we employ auxiliary subtasks with sizes $\tilde{N}$ drawn uniformly from $\tilde{N} \in \{N_{\min}, \dots, N_{\max}\}$ to evaluate the updates for the variational GMM parameters and the model parameters. Note that our algorithm does not require to sample context sets during meta training from the auxiliary subtasks. Furthermore, recall that we train one variational GMM for each auxiliary subtask and retain those GMMs over the whole course of meta training, so we fix a set of $\tilde{L}$ auxiliary subtasks at the beginning of meta-training (in contrast to standard NPs, which sample new subtasks for each minibatch).

We use the following settings for $N_{\min}$, $N_{\max}$ in our experiments: $N_{\min} = 1$, $N_{\max} = N$, except for MNIST image completion where we use $N_{\max} = N/2$. Further, we use $\tilde{L} = 32L$, except for MNIST image completion where we use $\tilde{L} = 8$.

B.3.3 Metrics

For each model-dataset combination, we retrain the best hyperparameter setting determined according to Sec. B.3.1 with 8 different random seeds used for model initialization, and report the median value together with $(5\%, 95\%)$ percentiles of the metrics computed according to the formulae provided below. For all experiments (except the MNIST image completion experiment), we evaluate all metrics on $L = 256$ unseen test tasks $\mathcal{D}_{1:L}$ with $\mathcal{D}_\ell = \left\{\boldsymbol{y}_{\ell,1:N}, \boldsymbol{x}_{\ell,1:N}\right\}$ and $N = 64$, from which we sample context sets $\mathcal{D}_\ell^c \subset \mathcal{D}_\ell$. For the image completion experiment we use $L = 1024$ and $N = 784$ (the number of pixels per image). We report the results in dependence of the context set size.

B.3.3.1 Log Marginal Predictive Likelihood (LMLHD)

For a given task ℓ the LMLHD is defined by Eq. (4.7), which we restate here for convenience:

$$\log q_\theta\left(\boldsymbol{y}_{\ell,1:N} \mid \boldsymbol{x}_{\ell,1:N}, \mathcal{D}_\ell^c\right) \equiv \log \int \prod_{n=1}^{N} p_\theta\left(\boldsymbol{y}_{\ell,n} \mid \boldsymbol{x}_{\ell,n}, \boldsymbol{z}_\ell\right) q\left(\boldsymbol{z}_\ell \mid \mathcal{D}_\ell^c\right) \mathrm{d}z_\ell. \tag{B.7}$$

Here, we use the generic notation $q\left(\boldsymbol{z}_\ell \mid \mathcal{D}_\ell^c\right)$ to denote the task posterior TP approximation, the concrete form of which depends on the BML model under consideration. As the integral is analytically intractable, we resort to an MC approximation. To this end, we sample $S = 1024$ samples $\boldsymbol{z}_{\ell,s} \sim q\left(\boldsymbol{z}_\ell \mid \mathcal{D}_\ell^c\right)$ in the test set and compute [Vol21]

$$\begin{aligned}\log q_\theta\left(\boldsymbol{y}_{\ell,1:N} \mid \boldsymbol{x}_{\ell,1:N}, \mathcal{D}_\ell^c\right) &\equiv \log \int \prod_{n=1}^{N} p_\theta\left(\boldsymbol{y}_{\ell,n} \mid \boldsymbol{x}_{\ell,n}, \boldsymbol{z}_\ell\right) q\left(\boldsymbol{z}_\ell \mid \mathcal{D}_\ell^c\right) \mathrm{d}z_\ell \\ &\approx \log \frac{1}{S} \sum_{s=1}^{S} \prod_{n=1}^{N} p_\theta\left(\boldsymbol{y}_{\ell,n} \mid \boldsymbol{x}_{\ell,n}, \boldsymbol{z}_{\ell,s}\right) \\ &= -\log S + \operatorname{logsumexp}_{s=1}^{S} \sum_{n=1}^{N} \log p_\theta\left(\boldsymbol{y}_{\ell,n} \mid \boldsymbol{x}_{\ell,n}, \boldsymbol{z}_{\ell,s}\right).\end{aligned} \tag{B.8}$$

where logsumexp denotes the numerically stable implementation of the function $\log \sum_s \exp(x_s)$, available in any scientific computing framework. We then compute the median of this expression over all tasks of the test set.

B.3.3.2 Mean Squared Error (MSE)

We report the MSE w.r.t. the mean prediction. That is, for a given task ℓ, we again draw $S = 1024$ samples $\boldsymbol{z}_{\ell,s} \sim q\left(\boldsymbol{z}_\ell \mid \mathcal{D}_\ell^c\right)$ and compute

$$\operatorname{MSE}\left(\boldsymbol{y}_{\ell,1:N}, \boldsymbol{x}_{\ell,1:N}\right) \equiv \frac{1}{N} \sum_{n=1}^{N} \left(\frac{1}{S} \sum_{s=1}^{S} \operatorname{dec}_\theta^\mu\left(\boldsymbol{x}_{\ell,n}, \boldsymbol{z}_{\ell,s}\right) - \boldsymbol{y}_{\ell,n}\right)^2. \tag{B.9}$$

We then compute the median of this expression over all tasks of the test set.

B.3.3.3 ELBO Looseness

For a given task ℓ, we define the ELBO looseness as the KL-divergence between the approximate task posterior $q\left(\boldsymbol{z}_\ell \mid \mathcal{D}_\ell\right)$ and the task posterior in the model $q_\theta\left(\boldsymbol{y}_{\ell,1:N} \mid \boldsymbol{x}_{\ell,1:N}, \mathcal{D}_\ell^c\right)$ Eq. (B.7), which we denote by $q_\theta\left(\boldsymbol{z}_\ell \mid \mathcal{D}_\ell, \mathcal{D}_\ell^c\right) \equiv \frac{p_\theta(\boldsymbol{y}_{\ell,1:N} \mid \boldsymbol{x}_{\ell,1:N}, \boldsymbol{z}_\ell) q(\boldsymbol{z}_\ell \mid \mathcal{D}_\ell^c)}{q_\theta(\boldsymbol{y}_{\ell,1:N} \mid \boldsymbol{x}_{\ell,1:N}, \mathcal{D}_\ell^c)}$. This decomposes as

$$\begin{aligned}\mathrm{KL}\left[q\left(\boldsymbol{z}_\ell \mid \mathcal{D}_\ell\right) \| q_\theta\left(\boldsymbol{z}_\ell \mid \mathcal{D}_\ell, \mathcal{D}_\ell^c\right)\right] &= \log q_\theta\left(\boldsymbol{y}_{\ell,1:N} \mid \boldsymbol{x}_{\ell,1:N}, \mathcal{D}_\ell^c\right) \\ &- \mathbb{E}_{q(\boldsymbol{z}_\ell \mid \mathcal{D}_\ell)}\left[\sum_{n=1}^{N} \log p_\theta\left(\boldsymbol{y}_{\ell,n} \mid \boldsymbol{x}_{\ell,n}, \boldsymbol{z}_\ell\right) + \log \frac{q\left(\boldsymbol{z}_\ell \mid \mathcal{D}_\ell^c\right)}{q\left(\boldsymbol{z}_\ell \mid \mathcal{D}_\ell\right)}\right].\end{aligned} \tag{B.10}$$

The second term is the ELBO,

$$\mathcal{L}\left(\boldsymbol{\theta}, \mathcal{D}_\ell^c, \mathcal{D}_\ell\right) \equiv \mathbb{E}_{q(\boldsymbol{z}_\ell|\mathcal{D}_\ell)}\left[\sum_{n=1}^{N} \log p_{\boldsymbol{\theta}}\left(\boldsymbol{y}_{\ell,n} \mid \boldsymbol{x}_{\ell,n}, \boldsymbol{z}_\ell\right) + \log \frac{q\left(\boldsymbol{z}_\ell \mid \mathcal{D}_\ell^c\right)}{q\left(\boldsymbol{z}_\ell \mid \mathcal{D}_\ell\right)}\right], \tag{B.11}$$

where we made its dependence on both the test set $\mathcal{D}_\ell$ and the context set $\mathcal{D}_\ell^c \subset \mathcal{D}_\ell$ explicit (in contrast to our notation in the main part of this paper). We say that the ELBO is tight if its looseness is zero. Then, $\log q_{\boldsymbol{\theta}}\left(\boldsymbol{y}_{\ell,1:N} \mid \boldsymbol{x}_{\ell,1:N}, \mathcal{D}_\ell^c\right) = \mathcal{L}\left(\boldsymbol{\theta}, \mathcal{D}_\ell^c, \mathcal{D}_\ell\right)$, and the optimization of the ELBO w.r.t. $\boldsymbol{\theta}$ is equivalent to the optimization of the LMLHD.

For our ablation study (Sec. 4.5.2), we estimate the looseness of the ELBO by computing the difference of an importance-weighted MC estimate with proposal distribution $q\left(\boldsymbol{z}_\ell \mid \mathcal{D}_\ell\right)$ of the LMLHD and an MC estimate of the ELBO Eq. (B.11), each with $S = 1024$ samples $\boldsymbol{z}_{\ell,s} \sim q\left(\boldsymbol{z}_\ell \mid \mathcal{D}_\ell\right)$.

B.4 Data Generation

We provide details on the meta-datasets we use to train the models we compare in Sec. 4.5. Concretely, we provide

- the dimension d_x of inputs $\boldsymbol{x}_{\ell,n} \in \mathbb{R}^{d_x}$,
- the domain $\mathcal{C} \subset \mathbb{R}^{d_x}$ from which we uniformly sample $\boldsymbol{x}_{\ell,n}$,
- the dimension d_y of targets $\boldsymbol{y}_{\ell,n} \in \mathbb{R}^{d_y}$,
- an expression for the function $f_\ell : \mathbb{R}^{d_x} \to \mathbb{R}^{d_y}$, s.t., $\boldsymbol{y}_{\ell,n} = f_\ell\left(\boldsymbol{x}_{\ell,n}\right) + \boldsymbol{\varepsilon}_n$,
- the noise standard deviation σ, s.t., $\boldsymbol{\varepsilon}_n \sim \mathcal{N}\left(0, \sigma^2\right)$,
- the number L of meta-tasks and the number N of datapoints for each meta-task.

We denote the uniform distribution on $(a, b)^d \subset \mathbb{R}^d$ by $\mathrm{U}\left(a, b\right)^d$.

Sinusoidal Functions:

- $d_x = 1$
- $\mathcal{C} = [-5.0, 5.0]$
- $d_y = 1$
- $f_\ell\left(x\right) = A_\ell \sin\left(x - \phi_\ell\right)$, $A_\ell \sim \mathrm{U}\left(0.1, 5.0\right)$, $\phi_\ell \sim \mathrm{U}\left(0.0, \pi\right)$
- $\sigma = 0.25$
- $L = 64$, $N = 16$

Mix of Affine and Sinusoidal Functions:

- $d_x = 1$
- $\mathcal{C} = [-5.0, 5.0]$
- $d_y = 1$
- $f_\ell^1(x) = a_\ell x + b_\ell$, $a_\ell \sim \mathrm{U}(-3.0, 3.0)$, $b_\ell \sim \mathrm{U}(-3.0, 3.0)$,
  $f_\ell^2(x) = A_\ell \sin(x - \phi_\ell)$, $A_\ell \sim \mathrm{U}(0.1, 5.0)$, $\phi_\ell \sim \mathrm{U}(0.0, \pi)$
  f_ℓ is given either by f_ℓ^1 or f_ℓ^2 with probability 0.5.
- $\sigma = 0.25$
- $L = 64, N = 16$

RBF-GP samples:

- $d_x = 1$
- $\mathcal{C} = [-2.0, 2.0]$
- $d_y = 1$
- f_ℓ is drawn from a Gaussian process prior with RBF kernel with lengthscale $l_\ell \sim \mathrm{U}(0.5, 1.0)$ and signal variance $s_\ell \sim \mathrm{U}(0.5, 1.0)$.
- $\sigma = 0.1$
- $L = 64, N = 16$

Forrester 1D:

- $d_x = 1$
- $d_y = 1$
- We use the parameterized Forrester function [For08] as defined on https://www.sfu.ca/~ssurjano/forretal08.html.
- $\sigma = 0.25$
- $L = 64, N = 16$

Branin 2D:

- $d_x = 2$
- $d_y = 1$
- We use the definition given on https://www.sfu.ca/~ssurjano/branin.html and apply translations $\boldsymbol{\tau}_\ell \sim \mathrm{U}(-0.25, 0.25)^2$ to $\boldsymbol{x}$, and scale the function values by $s_\ell \sim \mathrm{U}(0.75, 1.25)$.
- $\sigma = 0.25$
- $L = 64, N = 16$

Hartmann 3D:

- $d_x = 3$
- $d_y = 1$
- We use the definition given on https://www.sfu.ca/~ssurjano/hart3.html and apply translations $\boldsymbol{\tau}_\ell \sim \mathrm{U}\left(-0.25, 0.25\right)^3$ to $\boldsymbol{x}$, and scale the function values by $s_\ell \sim \mathrm{U}\left(0.75, 1.25\right)$.
- $\sigma = 0.1$
- $L = 64, N = 16$

4D Furuta Dynamics Prediction:

- $d_x = 4$
- $d_y = 4$
- We use the dynamics equations given in Cazzolato et al. [Caz11] to simulate episodes, starting from the pendulum balancing in the upright position. The input is the current system state $\boldsymbol{x} \in \mathbb{R}^4$, the target is the difference to the next system state $\boldsymbol{x}_{\text{next}} \in \mathbb{R}^4$, i.e., $\boldsymbol{y} = \Delta\boldsymbol{x} \equiv \boldsymbol{x}_{\text{next}} - \boldsymbol{x} \in \mathbb{R}^4$.
- Noise is generated by random actions on the joints.
- $L = 64, N = 64$

2D MNIST Image Completion:

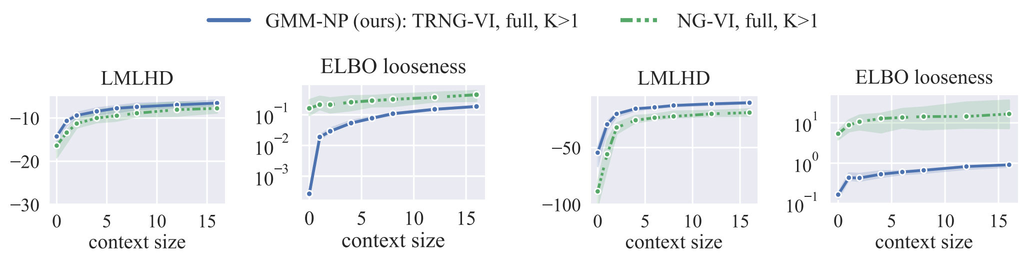
- $d_x = 2$
- $d_y = 1$
- We use the MNIST handwritten image database [LeC10]. Each image corresponds to one task. The input $\boldsymbol{x}$ is the pixel location, the target y is the pixel intensity.
- $\sigma = 0.25$
- $L = 60000, N = 784$

B.5 Further Experimental Results

We provide further experimental results for the experiments presented in Sec. 4.5.

B.5.1 Ablation: Trust Regions

In Fig. B.1 we compare two methods for step size control for natural gradient VI, namely direct step size control as proposed by [Lin20] and trust region step size control [Are23], as used by our GMM-NP algorithm. We observe that trust regions lead to more robust optimization of the variational parameters, and, thus, to tighter ELBOs. This allows more efficient optimization of the model parameters, leading to improved predictive performance.



(a) Sinusoidal functions. **(b)** Mix of affine and sinusoidal functions.

Figure B.1: Log marginal predictive likelihood (LMLHD) and ELBO looseness over context size for our trust region natural gradient VI (TRNG-VI)-based [Are23] GMM-NP algorithm in comparison to iBayes-GMM [Lin20] that uses direct step size control instead of trust regions (NG-VI). Trust regions improve variational optimization, leading to tighter ELBOs, and, consequently, to improved predictive performance.

B.5.2 Bayesian Optimization Experiments

We provide the full set of results for our Bayesian optimization experiments, cf. Sec. 4.5.3: Fig. B.2 shows the optimization regret for all four evaluated function classes, and Fig. B.3 the corresponding results for LMLHD and MSE.

B.5.3 2D Image Completion on MNIST

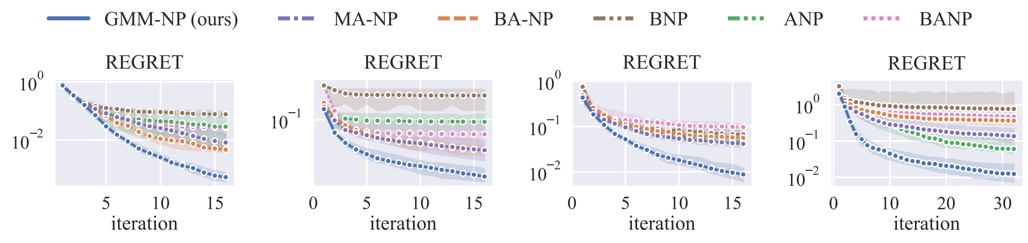
Fig. B.4 shows the full set of predictions on the 2D image completion experiment on MNIST.

B.5.4 Visualization of Model Predictions

Figs. B.5 and B.6 show further visualization of predictions of models trained on the mix of affine and sinusoidal functions.

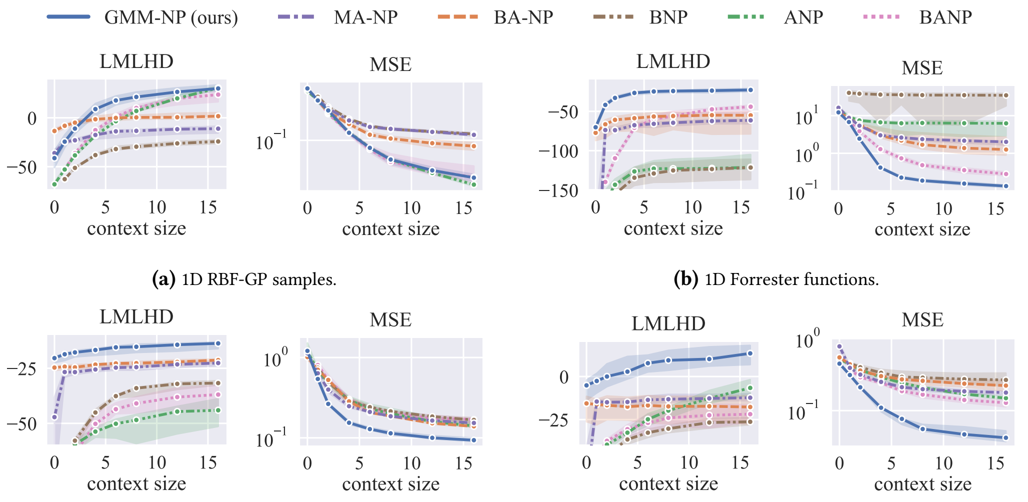
B.5.5 Visualization of Latent Space Structure

We provide further visualizations similar to Fig. 4.5 of the task posterior approximation and corresponding function samples of our GMM-NP, when trained on the sinusoidal function class.



(a) 1D RBF-GP samples. **(b)** 1D Forrester. **(c)** 2D Branin. **(d)** 3D Hartmann.

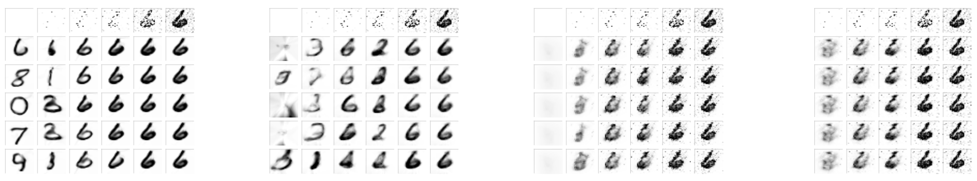
Figure B.2: Simple regret over optimization iteration, when using BML models as Bayesian Optimization (BO) surrogates on various function classes. As BO relies on well-calibrated uncertainty predictions, the results demonstrate that GMM-NP provides superior uncertainty estimates.



(a) 1D RBF-GP samples. **(b)** 1D Forrester functions.

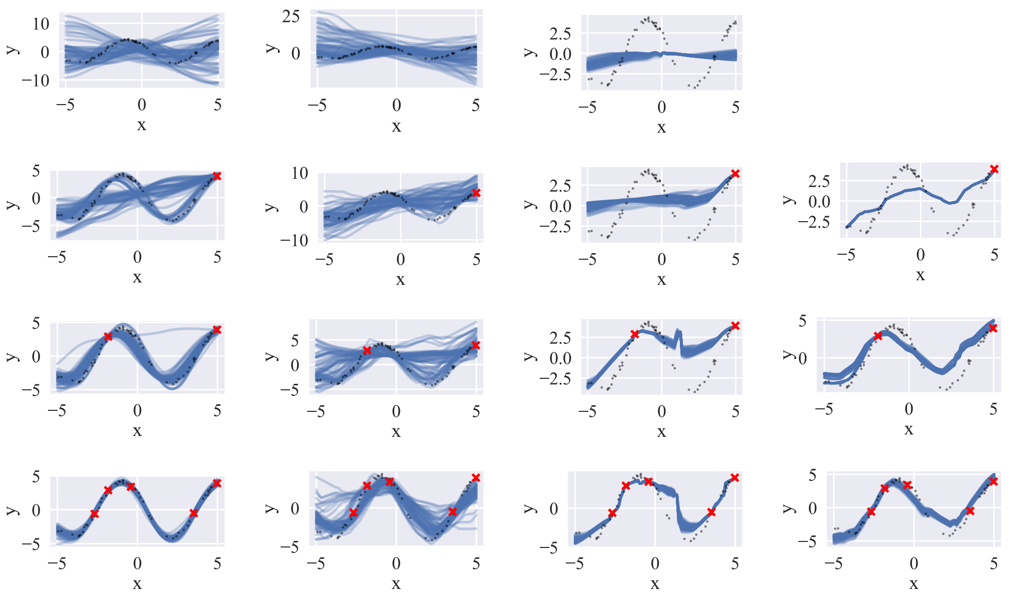
(c) 2D Branin functions. **(d)** 3D Hartmann-3 functions.

Figure B.3: Log marginal predictive likelihood (LMLHD) and mean squared error (MSE) over context size on various function classes. GMM-NP generally performs favorably, showing accurate predictions with well-calibrated uncertainties.



(a) GMM-NP (ours). **(b)** BA-NP. **(c)** ANP. **(d)** BANP.

Figure B.4: Predictions on an unseen instance of the MNIST 2D image completion task, showing the digit "6". The first row of each panel shows the context pixels (ranging from zero pixels in the left column to the full image in the right column). The remaining rows show five samples from the BML models, conditioned on the context pixels shown in the first row. The results are consistent with observations from the other experiments (e.g., Fig. B.5): our GMM-NP model shows highly variable samples for small context sets, yielding an accurate estimate of epistemic uncertainty, and contracts properly around the ground truth when more context information is available. BA-NP also shows variable samples, albeit of lower quality. ANP and BANP yield crisp predictions but massively overfit to the noise, explaining their low LMLHD scores. Note also that BANP does not allow predictions for empty context sets.



(a) GMM-NP (ours). **(b)** BA-NP. **(c)** ANP. **(d)** BANP.

Figure B.5: Function samples computed by various BML models (columns), trained on a function class consisting of a mix of affine and sinusoidal functions (cf. Sec. 4.5.1), when provided with increasing amounts of context examples (red crosses, rows) from an unseen sinusoidal representative function. We observe that our GMM-NP model accurately quantifies epistemic uncertainty through the variability of its function samples. BA-NP also shows variable samples, but does not achieve the same predictive performance due to its inaccurate approximation of the task posterior distribution. ANP and BANP, both of which employ deterministic computation paths with attention modules, produce essentially deterministic predictions that massively overfit the context data and fail to give a reasonable estimate of the predictive distribution. Therefore, these models have to quantify epistemic uncertainty through the likelihood noise variance, which is ineffective, cf. Fig. B.6. Note also that BANP does not provide predictions for empty context sets.

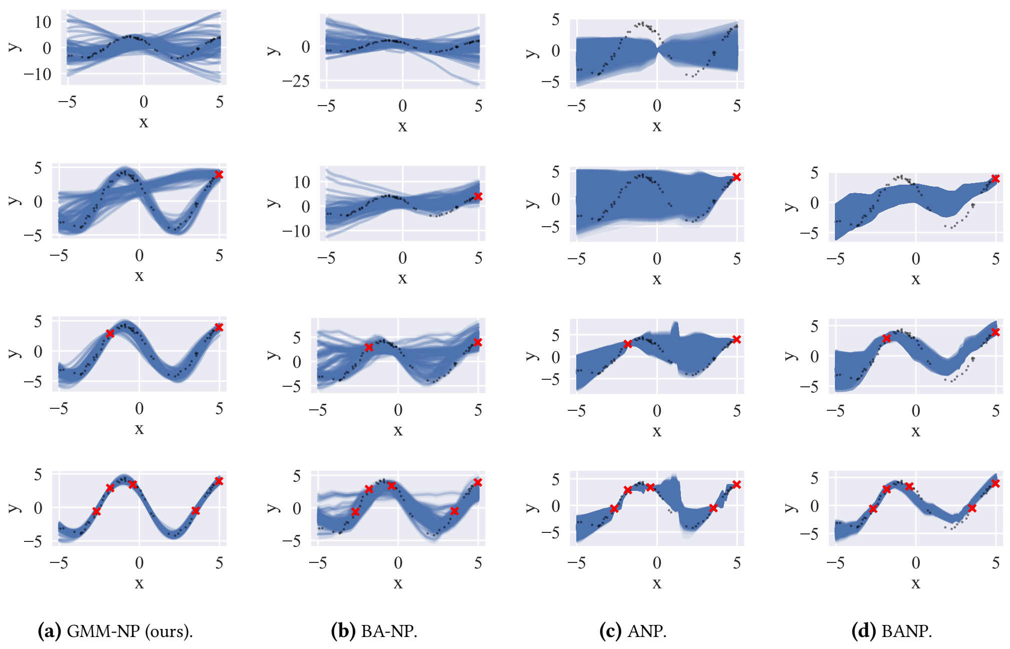
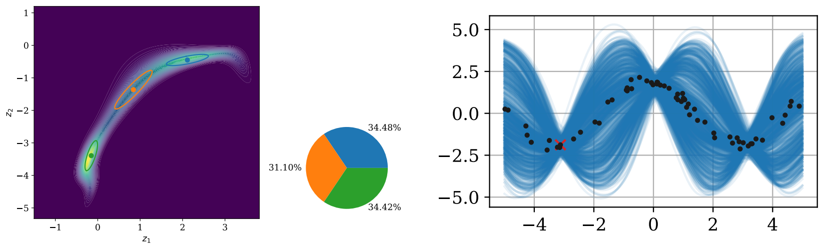
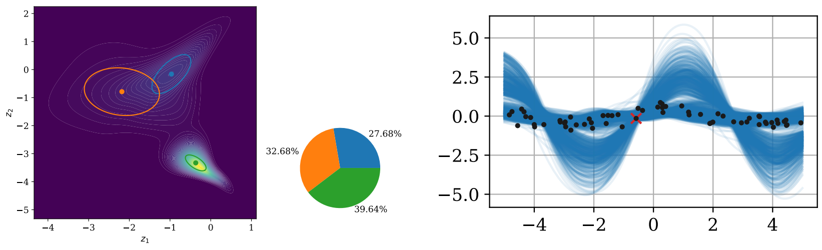


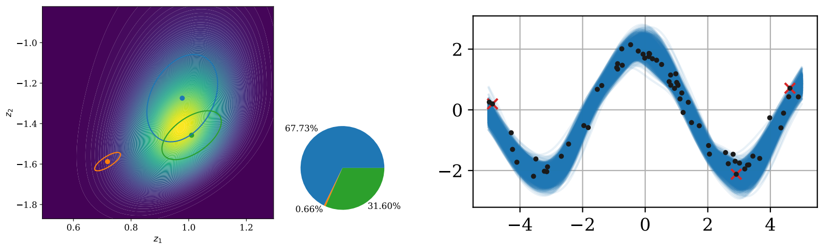
Figure B.6: This figure shows the same data as Fig. B.5, but for each function sample we also show a band of ± 1 standard deviation of the observation noise, as computed by the decoder DNN. GMM-NP quantifies epistemic uncertainty correctly through its task posterior approximation, and thus does not have to rely on the decoder DNN to quantify epistemic uncertainty through the observation noise. In contrast, ANP and BANP fail to produce variable function samples, and have to make up for that by quantifying epistemic uncertainty through the observation noise, which is ineffective. Note also that BANP does not provide predictions for empty context sets.



(a) A small context set (one single example indicated by the red cross) yields a highly correlated, multi-modal task posterior distribution. Our GMM approximation correctly captures this, s.t., amplitudes and phases of the predicted sinusoidal functions are in accordance with the observed context data point.



(b) A second example on another instance of the sinusoidal function class, where the task posterior shows pronounced multimodality, which translates into a bimodal predictive distribution.



(c) Larger context sizes (three examples, red crosses) leave less task ambiguity, resulting in a unimodal and nearly isotropic task posterior distribution. Our GMM approximation again correctly approximates this distribution, making use of only two of the $K = 3$ mixture components (the mixture weight of the orange component is close to zero, so no samples from this component are observed).

Figure B.7: Visualization of our GMM-NP model for a $d_z = 2$ dimensional latent space, trained on sinusoidal functions with varying amplitudes and phases, cf. Sec. 4.5.1. Left panels: unnormalized task posterior distribution (contours) and variational GMM approximation with $K = 3$ components (ellipses, mixture weights in %). Right panels: corresponding samples from our model (blue lines), when having observed a context data set (red crosses), together with unobserved ground truth data (black dots). The visualizations show that (i) the true task posterior distribution can be highly correlated and multimodal, i.p., for small context sets (panels a,b), (ii) our variational task posterior approximation correctly approximates this distribution, which (iii) leads to expressive predictive distributions that incorporate both the inductive priors learned from the meta-dataset (all samples are sinusoidal in shape) and the additional information contained in the context set (all samples pass close to the context data).

B.5.6 Runtime Comparison

B.5.6.1 Discussion of Limitations

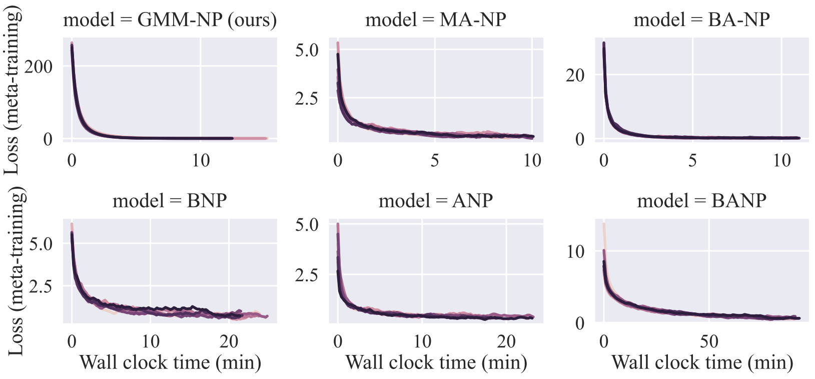
As the meta-training stage of GMM-NP requires computational effort comparable to standard NP (cf. Sec. 4.4), the only computational overhead of our algorithm occurs at test time, due to the optimization loop required to fit a variational GMM to $\mathcal{D}_*^c$. While this can be trivially parallelized for multiple test tasks, it incurs a higher computational burden in comparison to the single forward pass through NP's set encoder (we provide an evaluation of the runtime of our algorithm on the synthetic tasks studied in Sec. 4.5.1 below). We leave a detailed examination for future work, but mention two possible remedies: (i) for problems where test data arrives sequentially, we expect that a few update steps in $\boldsymbol{\phi}_*$ suffice to reach convergence, and (ii) it might be possible to find amortized approximations to Eqs. (4.13), similar in spirit to standard NP, that retain the advantages of TRNG-VI.

B.5.6.2 Meta-Training

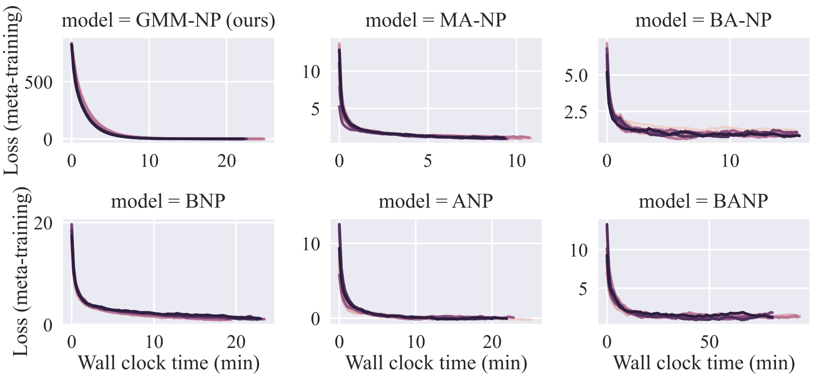
Fig. B.8 shows the learning curves for meta-training corresponding to the results presented in the main part of this paper. As discussed in Sec. 4.4, GMM-NP incurs a computational cost comparable to the baseline methods.

B.5.6.3 Test-time Adaptation

As discussed in Sec. 4.4, GMM-NP does not amortize TP inference, i.e., it does not learn a set encoder architecture, but adapts new variational GMMs at test time. Naturally, this incurs a higher computational cost in comparison to amortized architectures, which compute predictions on test tasks in a single forward pass through their set-encoder – decoder architecture. In Fig. B.9, we show the learning curves for fitting variational GMMs (by iterating Eqs. (4.13)) to the test tasks and for the range of context sizes used to compute the results presented in Sec. 4.5.1. GMM-NP's TRNG-VI optimization converges in approximately $0.1\,\mathrm{s} - 1\,\mathrm{s}$ per test task (depending on the context set size).

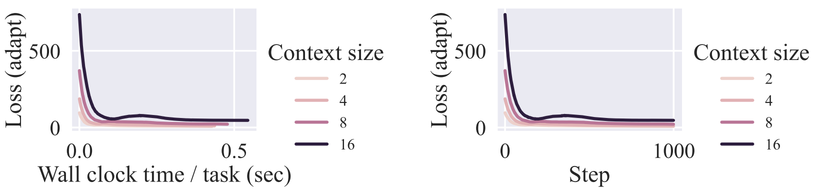


(a) Learning curves on the sinusoidal function class.

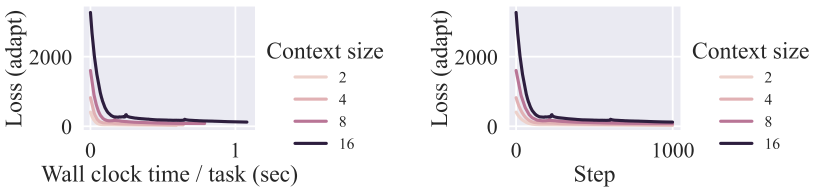


(b) Learning curves on the mix of affine and sinusoidal function class.

Figure B.8: Learning curves for meta-training on the synthetic datasets, cf. Sec. 4.5.1. For each method, we show the learning curves for the 8 seeds used to compute the results presented in the main text. For GMM-NP, we show the loss for the decoder parameters $\boldsymbol{\theta}$, for the other methods we show the joint loss for the encoder and decoder parameters $(\boldsymbol{\phi}, \boldsymbol{\theta})$. Note that for GMM-NP, convergence of $\boldsymbol{\theta}$ implies convergence of the variational parameters $\boldsymbol{\phi}$. As discussed in Sec. 4.4, GMM-NP incurs a computational cost comparable to the baseline methods.



(a) Sinusoidal functions.



(b) Mix of affine and sinusoidal functions.

Figure B.9: Learning curves for fitting variational GMMs to the test tasks by TRNG-VI [Are23], as used by our GMM-NP (Sec. 4.4), on the synthetic datasets (Sec. 4.5.1). The quantity labelled "Loss (adapt)" is the expected negative log density of the unnormalized TP under the GMM TP approximation. Note that this is not the loss function optimized by iterating Eqs. (4.13), but it serves as a proxy to judge convergence. We show results in terms of wall clock time per test task (left panels) and in terms of TRNG-VI steps (right panels), for the range of context sizes used to compute the results in the main text. GMM-NP's TRNG-VI optimization converges in approximately 0.1 s – 1 s per test task (depending on the context set size).

B.5.7 Analysis of HPO results

As discussed in Secs. 4.5 and B.3, we optimized architectural hyperparameters individually for each model-dataset combination presented in our empirical evaluation, in order to arrive at a fair comparison of our GMM-NP with the baseline methods. In Tab. B.2, we provide the resulting settings for the latent dimensionality d_z, and the number of parameters of the BML models compared in Sec. 4.5.1. While the number of variational parameters during meta-training is naturally comparably high for non-amortizing methods such as GMM-NP, we observe that the expressive GMM-NP TP approximation allows comparably lightweight decoders and small latent dimensions. This is intuitive, as simple TP approximations require (i) large latent dimensions to encode relevant information in the latent space, together with (ii) expressive decoder architectures to transform the simple latent distribution into an expressive predictive distribution. Note further that the variational parameters belonging to different tasks are not coupled in non-amortizing architectures such as ours, which allows trivial parallelization of the variational optimization between tasks, explaining why the computational cost is easily manageable, cf. Sec. B.5.6. Note also that the number of variational parameters one has to store and adapt for GMM-NP to make predictions on unseen test tasks is comparably small because the variational GMMs learned during meta-training can be discarded as they are not required for predictions at test time.

Table B.2: Results of our hyperparameter optimization on the sinusoidal function class and on the mix of affine and sinusoidal functions. We provide the settings for the latent dimensionality d_z and the number of parameters of the BML models compared in Sec. 4.5.1 (i.e., the number of decoder parameters $|\theta|$ as well as the number of encoder / variational parameters $|\phi|$). If attentive modules are present, their parameters are counted as being part of the encoder. For our GMM-NP, we also provide the number of GMM-components K. Furthermore, as GMM-NP does not amortize TP-inference but learns separate variational GMMs for each subtask generated from the meta-dataset (cf. Secs. 4.4 and B.3.2), we also provide the total number of variational GMM parameters during meta-training. Note that these variational GMMs are decoupled and can be optimized in parallel. Furthermore they are not required for predictions at test time and can be discarded after meta-training.

		d_z	K	$\|\theta\|$	$\|\phi\|$ (per task)	$\|\phi\|$ (meta-training)
Sinusoid	GMM-NP (ours)	3	4	121	39	79872
	MA-NP	10	-	2298	2595	2595
	BA-NP	27	-	7334	7386	7386
	BNP	56	-	12770	19488	19488
	ANP	19	-	4096	10335	10335
	BANP	40	-	6562	28320	28320
Line-Sine	GMM-NP (ours)	4	4	2289	59	120832
	MA-NP	19	-	2562	3679	3679
	BA-NP	30	-	11592	11650	11650
	BNP	54	-	11882	18144	18144
	ANP	30	-	4496	14448	14448
	BANP	32	-	4226	18304	18304

C Meta-Learning Acquisition Functions for Bayesian Optimization

C.1 Additional Experimental Results

C.1.1 Interpretation of Neural AF Search Strategies

We provide additional experimental results to demonstrate that MetaBO's neural AFs learn representations that go beyond some kind of standard AF combined with a prior over $\mathcal{D}$.

C.1.1.1 Emergence of Non-Greedy Search Strategies

To obtain intuition about the kind of search strategies MetaBO is able to learn, we devised two classes of one-dimensional toy objective functions.

The first class of objective functions (Rhino-1, cf. Fig. C.1) is generated by applying random translations sampled uniformly from $t \in [-0.2, 0.2]$ to a function which is given by the superposition of two Gaussian bumps with different heights and widths and fixed distance,

$$f_{\mathrm{R1}}(x,t) \equiv 0.5 \cdot \mathcal{N}(x \mid \mu = 0.3 - t, \sigma = 0.1) + 3.0 \cdot \mathcal{N}(x \mid \mu = 0.7 - t, \sigma = 0.01)\,, \tag{C.1}$$

where we define $\mathcal{N}(x \mid \mu, \sigma) \equiv \exp(-1/2 \cdot (x-\mu)^2/\sigma^2)$. The second class of objective functions (Rhino-2, cf. Fig. C.2) is given by uniformly sampling the parameter $h \in [0.6, 0.9]$ of the function

$$f_{\mathrm{R2}}(x,h) \equiv h \cdot \mathcal{N}(x \mid \mu = 0.2, \sigma = 0.1) + 2.0 \cdot \mathcal{N}(x \mid \mu = h, \sigma = 0.01) - 1.0. \tag{C.2}$$

For both of these function classes it is intuitively clear that the optimal search strategy involves a first non-greedy evaluation to identify the specific instance of the target function. Indeed, for all instances of these function classes, the smaller and wider bumps overlap and encode information about the position of the sharp global optimum. Therefore, an optimal strategy spends the first evaluation at a fixed position x_0 where all smaller and wider bumps have non-negligible heights y_0. Then, for both function classes, the global optimum x^* can be determined exactly from y_0 (if we assume noiseless evaluations), such that x^* can be found in the second step. Figs. C.1, C.2 show that MetaBO indeed learns such non-greedy optimization strategies, which go far beyond a simple combination of a prior over $\mathcal{D}$ with some kind of standard AF. As mentioned in the main part of this paper, we suppose that MetaBO employs similar strategies on more complex function classes. For instance, we observe in the experiments on the global optimization benchmark functions (Fig. 5.3) that MetaBO consistently starts with higher regret than the pre-informed TAF which suggests that it learned to spend a few non-greedy evaluations at the beginning of an optimization run to identify the specific instance of the target function.

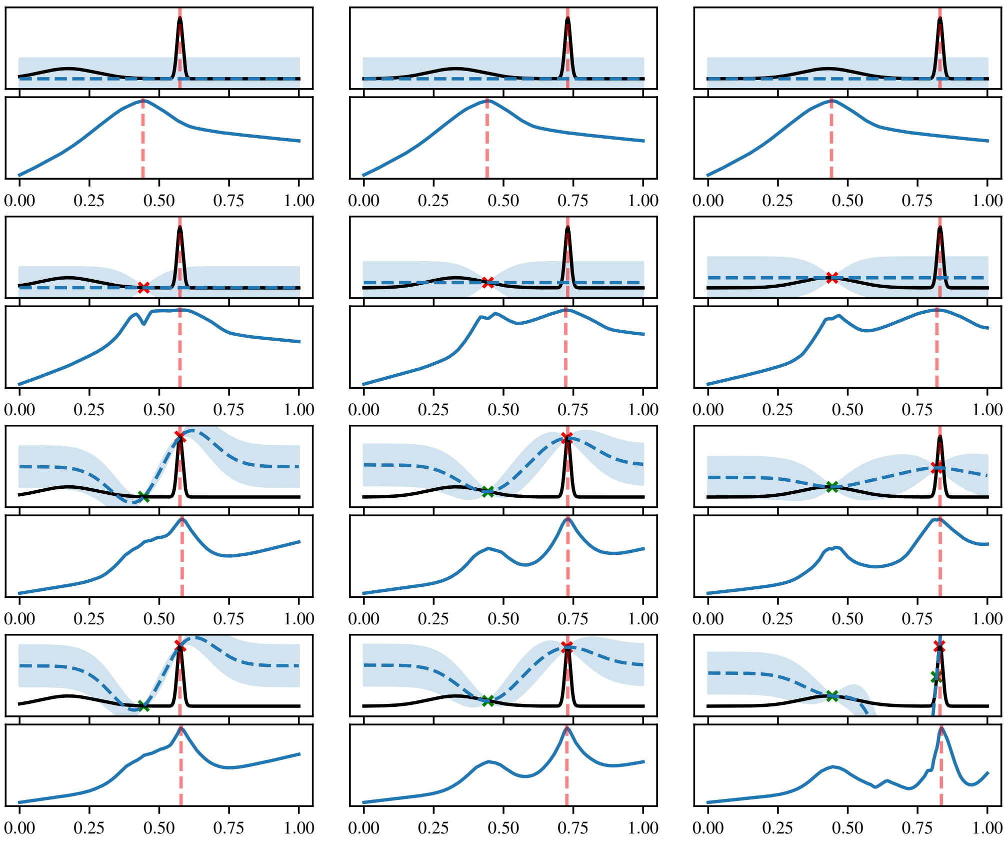


Figure C.1: Visualization of three BO episodes with neural AFs on the $1D$ Rhino-1 task. Each column of this figure correspond to one episode with three optimization steps. The uppermost row corresponds to the prior state before the objective function was queried. The fourth row depicts the state after three evaluations. Each subfigure shows the GP mean (dashed blue line), GP standard deviation (blue shaded area), and the ground truth function (black) in the upper panel as well as the neural AF in the lower panel. Dashed red lines indicate the maxima of the ground truth function and of the neural AF. Red and green crosses indicate the recorded data (the red cross corresponds to the most recent data point). Each instance of this task is generated by randomly translating an objective function with two peaks of different heights and widths. The distance between the local and global optimum is the same for each instance. MetaBO learns a sophisticated sampling strategy, spending a non-greedy evaluation at the beginning of each episode at a position where the smaller but wider peaks overlap for every instance of the function class to gain information about the location of the global optimum. Using this strategy, MetaBO is able to find the global optimum very efficiently.

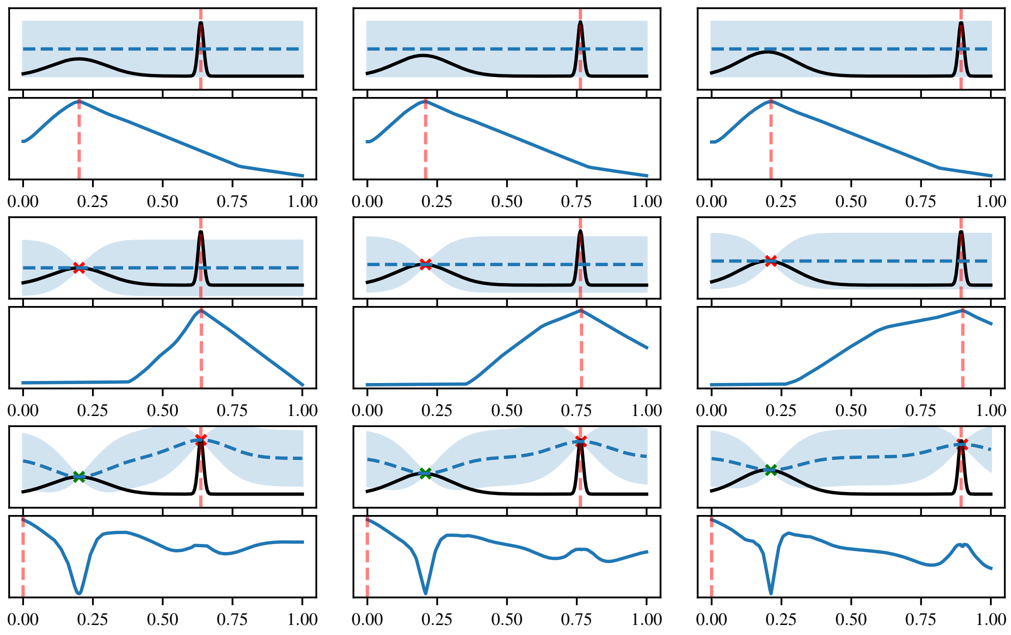


Figure C.2: Visualization of three episodes from the $1D$ Rhino-2 task. Each column of this figure correspond to one episode with two optimization steps. The uppermost row corresponds to the prior state before the objective function was queried. The third row depicts the state after two evaluations. Each subfigure shows the GP mean (dashed blue line), GP standard deviation (blue shaded area), and the ground truth function (black) in the upper panel as well as the neural AF in the lower panel. Dashed red lines indicate the maxima of the ground truth function and of the neural AF. Red and green crosses indicate the recorded data (the red cross corresponds to the most recent data point). Each instance of this task is generated by sampling the height h of a wide bump at a fixed location $x = 0.2$ and placing a sharp peak at $x = h$. MetaBO learns a sophisticated sampling strategy, spending a non-greedy evaluation at $x \approx 0.2$ at the beggining of each episode to gain information about the location of the global optimum. Using this strategy, MetaBO is able to find the global optimum very efficiently.

C.1.1.2 Additional Baseline Methods

To provide further evidence that MetaBO's neural AFs learn representations that go beyond a simple prior over $\mathcal{D}$ combined with some kind of standard AF, we show results for two additional baseline AFs which rely on such a naive combination.

We define the AF GMM-UCB as the following convex combination of a Gaussian Mixture Model (GMM) and the standard AF UCB:

$$\mathrm{GMM} - \mathrm{UCB}\,(x) \equiv w \cdot \mathrm{GMM}\,(x) + (1 - w) \cdot \mathrm{UCB}\,(x)\,. \tag{C.3}$$

The GMM is defined to have n_{comp} components and is fitted to the best designs from each of the M source tasks. Further, UCB is defined as

$$\mathrm{UCB}\,(x) \equiv \mu(x) + \beta\sigma(x)\,, \tag{C.4}$$

and we choose $\beta = 2$ as is common in BO.

Furthermore, we define EPS-GREEDY as the AF which, in each optimization step, samples without replacement from the set of best designs of each of the source tasks with probability ϵ, and uses standard EI with probability $1 - \epsilon$.

Note that these baseline methods are similar in spirit to the TAF-approach evaluated in the main part of this paper. Indeed, TAF, GMM-UCB, and EPS-GREEDY all rely on some kind of prior over $\mathcal{D}$ determined using the source data which is combined through a weighted superposition with some standard AF. However, TAF uses more principled methods (TAF-ME, TAF-R) to adaptively determine the weights of this superposition.

To obtain optimal performance of GMM-UCB and EPS-GREEDY, we chose the parameters for these methods by grid search on the test set[1] w.r.t. the median simple regret summed from $t = 0$ to $t = T = 30$. To tune w for GMM-UCB we tested 10 linearly spaced points in $[0.0, 1.0]$ as well as a schedule which reduces w from 1.0 to 0.0 over the course of one episode. Furthermore, we tested numbers of GMM-components $n_{\text{comp}} \in \{1, 2, 3, 4, 5\}$. Similarly, for EPS-GREEDY we tested ϵ on 10 linearly spaced points in $[0.0, 1.0]$ and also evaluated a schedule which reduces ϵ from 1.0 to 0.0 over an episode.

In Fig. C.3 we display the performance of GMM-UCB and EPS-GREEDY on the global optimization benchmark functions Branin, Goldstein-Price, and Hartmann-3 with the optimal parameter configurations (cf. Tab. C.1) and with $M = 50$ source tasks. MetaBO outperforms both GMM-UCB and EPS-GREEDY, which provides additional evidence that neural AFs learn representations which go beyond a simple combination of standard AFs with a prior over $\mathcal{D}$.

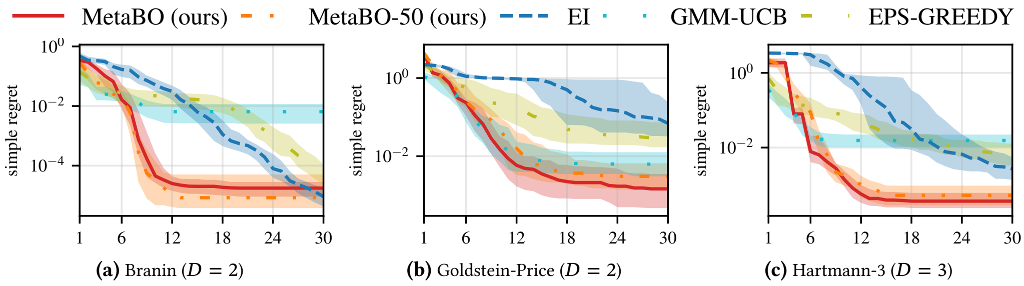


Figure C.3: Performance on three global optimization benchmark functions with random translations sampled uniformly from $[-0.1, 0.1]^D$ and scalings from $[0.9, 1.1]$. We present results for two additional baseline methods (GMM-UCB, EPS-GREEDY) which rely on a weighted superposition of a prior over $\mathcal{D}$ obtained from $M = 50$ source tasks and a standard AF and can thus be easily interpreted. As MetaBO produces more sophisticated search strategies, these approaches are not able to surpass MetaBO's performance.

[1] This yields an upper bound on the possible performance of GMM-UCB and EPS-GREEDY, as in practice one would have to estimate the parameters using a separate validation set.

Table C.1: Optimal parameters of GMM-UCB and EPS-GREEDY (determined on the test set).

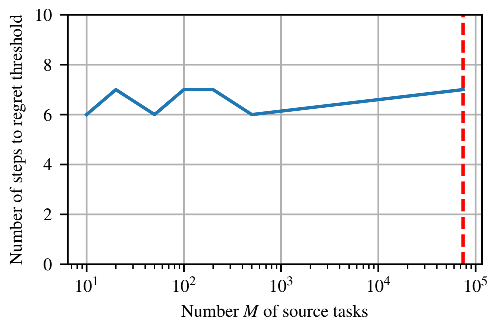
	w	n_{comps}	ϵ
Branin	0.22	3	linear schedule
Goldstein-Price	0.22	1	0.55
Hartmann-3	0.11	2	linear schedule

C.1.2 Dependence on the Number of Source Tasks

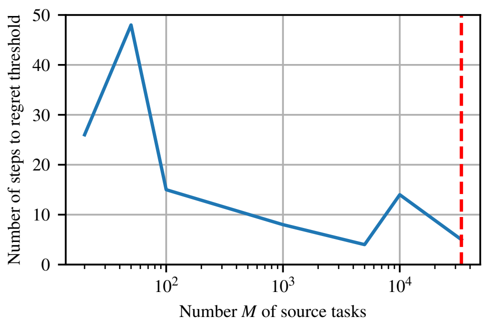
We argued in the main part of this paper that one main advantage of MetaBO over existing transfer learning methods for BO is its ability to process a very large amount of source data because it does not store all available data in GP models (in contrast to TAF) but rather accumulates the data in the neural AF weights. For tasks where source data are abundant (e.g., when they come from simulations, cf. Fig. 5.4), this frees the user from having to select a small subset of representative source tasks by hand, which can be intricate or even impossible for complex tasks. In addition, we showed in our experiments that MetaBO's applicability is not restricted to such cases, but that it also performs favourably with the same amount of source data as presented to the baseline methods on tasks which do not require a very large amount of source data to be solved efficiently (cf. Figs. 5.3, 5.5).

In Fig. C.4 we provide further evidence for this aspect by plotting the performance of MetaBO for different numbers M of source tasks on the Branin function and on functions from the simulation of the Furuta pendulum stabilization task. The results indicate that on the Branin function a small number of source tasks is already sufficient to obtain strong optimization performance. In contrast, the more complex stabilization task requires a much larger amount of source data to be solved reliably.

We emphasize that MetaBO's evaluation runtime does not depend on the number M of source tasks because a neural AF evaluation only requires one forward pass through a neural AF of fixed size. Therefore, it scales well to the regime of abundant source data. In contrast, TAF-ME's runtime scales linearly in the number M of source tasks and quadratically in the number N of data points per source task, while TAF-R shows an even stronger dependence on M due to the computation of the pairwise ranks. We underline this scaling behavior by presenting measured evaluation runtimes in Tab. C.2.



(a) Branin function



(b) Furuta pendulum in simulation

Figure C.4: Dependence of MetaBO's performance on the number of source tasks provided during training on the Branin function (cf. Fig. 5.3a) and on the stabilization task for the Furuta pendulum in simulation (cf. Fig. 5.4a). We show the number of steps MetaBO requires to reach a given performance in terms of median regret over 100 test functions in dependence of the number M of source tasks. As in the main part of this paper, we chose a constant budget of $T = 30$ on the Branin function and of $T = 50$ on the stabilization task. The dashed red line indicates the number of source tasks seen by the full version of MetaBO (a new function is sampled from the training distribution at the beginning of each optimization episode) at the point of convergence of meta-training. For the Branin function we chose the regret threshold $R = 10^{-3}$, which corresponds to the median final performance of TAF after $t = 30$ steps as presented in the main part of this paper (Fig. 5.3a). For the Furuta stabilization task, we chose the regret threshold $R = 1.0$, which corresponds approximately to the regret that has to be reached in simulation to allow stabilization on the real system. The results show that on the Branin function already a small number of source tasks is enough to obtain a powerful optimization strategy. In contrast, neural AFs trained on the more complex simulation-to-real task benefit from MetaBO's ability to process a very large amount of source tasks.

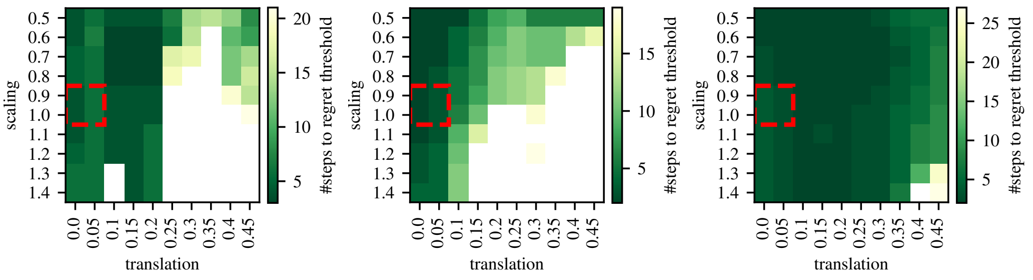
Table C.2: Comparison of evaluation runtimes per BO episode with budget $T = 30$ in s for various AFs, averaged over 10 BO runs. We show MetaBO's runtime for $M = 50$ source tasks as well as for the full version (where a new function is sampled from the training distribution in each BO run). For TAF, we indicate M and the number N of data points per source task by TAF-ME-M-N and TAF-R-M-N. Note that the absolute figures of the reported runtimes obviously depend on the hardware architecture used for the evaluation.

	Branin	Goldstein-Price	Hartmann-3
EI	0.13	0.13	0.16
MetaBO-50	0.60	0.55	0.82
MetaBO-full	0.62	0.59	0.81
TAF-ME-50-100	13	14	24
TAF-ME-50-200	29	28	35
TAF-ME-100-100	17	19	30
TAF-ME-100-200	47	49	65
TAF-R-50-100	50	50	56
TAF-R-50-200	61	60	69
TAF-R-100-100	100	100	110
TAF-R-100-200	120	120	140

C.1.3 Generalization Behavior

As described in the main part of this paper, MetaBO's primary use case is transfer learning, i.e., to speed up optimization on target functions similar to the source objective functions. Put differently, we are mainly interested in MetaBO's performance on unseen functions drawn from the training distribution. Nevertheless, studying MetaBO's generalization performance to functions outside of the training distribution can give interesting insights into the nature of the tasks we considered in the main part. Therefore, we present a study of MetaBO's generalization performance on the global optimization benchmark functions (Fig. C.5) as well as on the simulation-to-real experiment (Fig. C.6).

The results on the simulation-to-real task show that the neural AF generalizes better to heavy and long than to lightweight and short pendula. We suppose that this result is related to the fact that lightweight and short pendula show much faster dynamics due to their small moments of inertia than heavier and longer ones and are thus much harder to stabilize. Put more precisely, the change of the optimization landscape is much more pronounced when moving to lighter and smaller pendula than in the other direction. Similar conclusions can be drawn for the translated and scaled global optimization benchmark functions.



(a) Branin ($D = 2$) **(b)** Goldstein-Price ($D = 2$) **(c)** Hartmann-3 ($D = 3$)

Figure C.5: Generalization of neural AFs to functions outside of the training distribution (translations $t \in [-0.1, 0.1]$, scalings $s \in [0.9, 1.1]$, red square) on Branin, Goldstein-Price, and Hartmann-3. We evaluated the neural AFs on 100 test distributions with disjoint ranges of translations and scalings, each corresponding to one tile of the heatmap. The x- and y-labels of each tile denote the lower bounds of the translations t and scalings s of the respective test distribution from which the parameters were sampled uniformly (for each dimension we sampled the translation and its sign independently). The color encodes the number of optimization steps required to reach a given regret threshold. White tiles indicate that this threshold could not be reached withtin $T = 30$ optimization steps. The regret threshold was fixed for each function separately: we set it to the 1%-percentile of the set of regrets corresponding to function evaluations on a Sobol grid of one million points in the domain of the original objective functions.

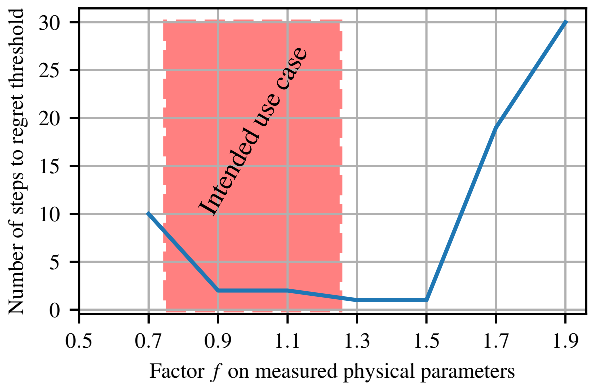


Figure C.6: Generalization of neural AFs to functions outside of the training distribution (75% to 125% of measured physical parameters, red square) on the simulation-to-real task. We evaluated neural AFs on test distributions with disjoint ranges of physical parameters (masses and lengths of the pendulum and arm). We sampled each physical parameter p_i uniformly on $[f \cdot p_{i,\text{measured}}, (f + 0.2) \cdot p_{i,\text{measured}}]$. Therefore, $f = 0.9$ corresponds to the interval containing the measured parameters. We plot f on the x-axis and the number of steps required to reach a regret threshold of $R = 1.0$ on the y-axis. Following our experience, this corresponds approximately to the regret that has to be reached in simulation to allow stabilization on the real system. We emphasize that the intended use case of MetaBO is on systems inside of the training distribution marked in red, as this distribution is chosen such that the true parameters are located inside of it with high confidence when taking into account the measurement uncertainty. Note that for small f the system becomes very hard to stabilize (lightweight and short pendula) such that the optimization landscape differs significantly from the training distribution, which is why the regret threshold cannot be reached within 30 steps for $f \leq 0.5$.

C.1.4 Full Set of Results from Main Part

C.1.4.1 Global Optimization Benchmark Functions

We provide the full set of results for the experiment on the global optimization benchmark functions. In Fig. C.7 we also include results for TAF with M = 20, showing that TAF's performance does not necessarily increase with more source data.

C.1.4.2 Simulation-to-Real Experiment

We provide the full set of results for the experiment on the global optimization benchmark functions, including the results for TAF-50, cf. Fig. C.8.

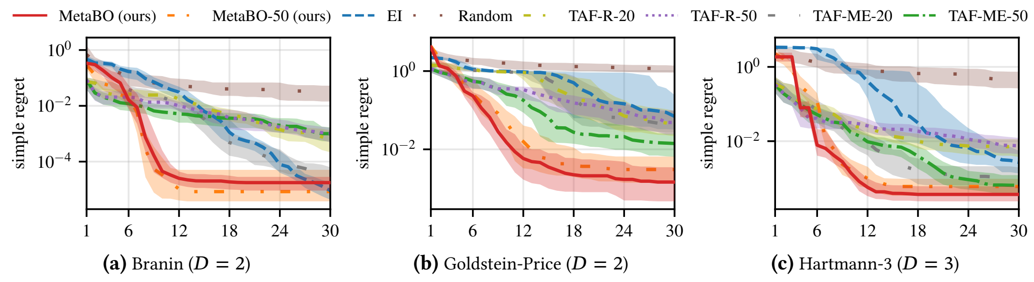


Figure C.7: Performance on three global optimization benchmark functions with random translations sampled uniformly from $[-0.1, 0.1]^D$ and scalings from $[0.9, 1.1]$. To test TAF's performance, we randomly picked M source tasks from this function class and evaluated both the ranking-based version (TAF-R-M) and the mixture-of-experts version (TAF-ME-M). We show results for $M \in \{20, 50\}$. Note that TAF's performance does not necessarily increase with more source data. We trained MetaBO on the same set of source tasks as TAF-50 (MetaBO-50). In contrast to TAF, MetaBO can also be trained without manually restricting the set of available source tasks. The corresponding results are labelled "MetaBO". MetaBO outperformed EI by clear margin, especially in early stages of the optimization. After few steps used to identify the specific instance of the objective function, MetaBO also outperforms both flavors of TAF over wide ranges of the optimization budget.

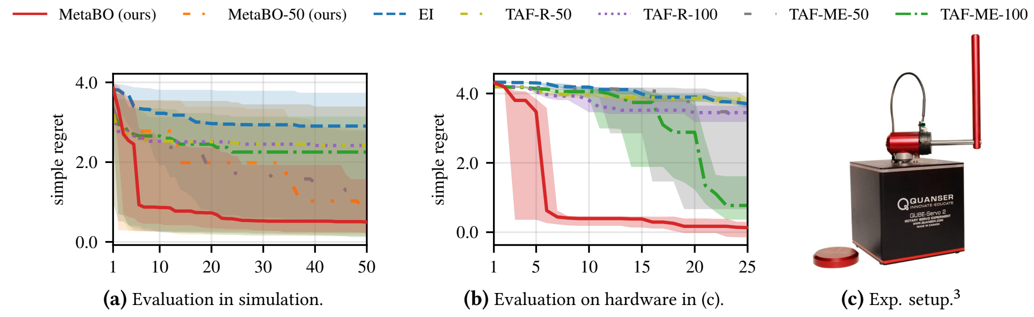


Figure C.8: Performance on a simulation-to-real task (cf. text). MetaBO and TAF used source data from a cheap numerical simulation. (a) Performance on an extended training set in simulation. (b) Transfer to the hardware depicted in (c), averaged over ten BO runs. MetaBO learned robust neural AFs with very strong early-time performance and online adaption to the target objectives, which reliably yielded stabilizing controllers after less than ten BO iterations while TAF-ME-50, TAF-ME-100, TAF-R-50, TAF-R-100, and EI explore too heavily. Comparing the results for MetaBO and MetaBO-50 in simulation, we observe that MetaBO benefits from its ability to learn from the whole set of available source data, while TAF's applicability is restricted to a comparably small number of source tasks.

C.2 Experimental Details

To foster reproducibility, we provide a detailed explanation of the settings used in our experiments and make source code available online.[1]

[1] https://github.com/boschresearch/MetaBO

C.2.1 General Implementation Details

In what follows, we explain all hyperparameters used in our experiments and summarize them in Tab. C.3. We emphasize that we used the same MetaBO hyperparameters for all our experiments, making our method easily applicable in practice.

Table C.3: Parameters of the MetaBO framework used in our experiments.

Description	**Value in experiments**
BO/AF parameters	
Cardinality N_{MS} of multistart grid	
Branin, Goldstein-Price	1000
Hartmann-3	2000
Simulation-to-real	10000
GPs ($D = 1, 2, 3, 4, 5$)	$500, 1000, 2000, 3000, 4000$
Cardinality N_{LS} of local search grid	N_{MS}
Number k of multistarts	5
MetaBO parameters	
Cardinality of ξ_{global}	N_{MS}
Cardinality of $\xi_{\text{local},t}$	k
Neural AF architecture	200 - 200 - 200 - 200, relu activations
PPO parameters [Sch17]	
Batch size	1200
Number of epochs	4
Number of minibatches	20
Adam learning rate	$1 \cdot 10^{-4}$
CPI-loss clipping parameter	0.15
Value network architecture	200 - 200 - 200 - 200, relu activations
Value coefficient in loss function	1.0
Entropy coefficient in loss function	0.01
Discount factor γ	0.98
GAE-λ [Sch16b]	0.98

C.2.1.1 Gaussian Process Surrogate Models

We used the implementation GPy [GPy12] with squared-exponential kernels (Matern-5/2 kernels for the corresponding experiments on general function classes) with automatic relevance determination and a Gaussian noise model and tuned the corresponding hyperparameters (noise variance, kernel lengthscales, kernel signal variance) offline by fitting a GP to the objective functions in the training and test sets using type-2 maximum likelihood. We also used the resulting hyperparameters for the source GPs of TAF. We emphasize that our method is fully compatible with other (online) hyperparameter optimization techniques, which we did not use in our experiments to arrive at a consistent and fair comparison with as few confounding factors as possible.

C.2.1.2 Baseline AFs

As is standard, we used the parameter-free version of EI. For TAF, we follow Wistuba et al. [Wis18] and evaluate both the ranking-based (TAF-R) as well as the product-of-experts (TAF-ME) versions. We detail the specific choices for the number of source tasks M and the number of datapoints N_{TAF} contained in each source GP in the main part of this paper.

For EI we used the midpoint of the optimization domain $\mathcal{D}$ as initial design. For TAF we did not use an initial design as it utilizes the information contained in the source tasks to warmstart BO. Note that MetaBO also works without any initial design.

C.2.1.3 Maximization of the AFs

Our method is fully compatible with any state-of-the-art method for maximizing AFs. In particular, our neural AFs can be optimized using gradient-based techniques. We chose to switch off any confounding factors related to AF maximization and used a hierarchical gridding approach for all evaluations as well as during training of MetaBO. For the experiments with continuous domains $\mathcal{D}$, i.e., all experiments except the HPO task, we first put a multistart Sobol grid with N_{MS} points over the whole optimization domain and evaluated the AF on this grid. Afterwards, we implemented local searches from the k maximal evaluations via centering k local Sobol grids with N_{LS} points, each spanning approximately one "unit cell" of the multistart grid, around the k maximal evaluations. The AF maximum is taken to be the maximal evaluation of the AF on these k Sobol grids. For the HPO task, the AF maximum can be determined exactly because the domain is discrete.

C.2.1.4 Reinforcement Learning Method

We use the trust-region policy gradient method Proximal Policy Optimization (PPO) [Sch17] as the algorithm to train the neural AF.

C.2.1.5 Reward Function

If the true maximum of the objective functions is not known at training time, we compute R_t with respect to an approximate maximum and define the reward to be given by $r_t \equiv -R_t$. This is the case for the experiment on general function classes (GP samples) where we used grid search to approximate the maximum as well as for the simulation-to-real task on the Furuta pendulum where we used the performance of a linear quadratic regulator (LQR) controller as an approximate maximum. For the experiments on the global optimization benchmark functions as well as on the HPO tasks, we do know the exact value of the global optimum. In these cases, we use a logarithmic transformation of the simple regret, i.e., $r_t \equiv -\log_{10} R_t$ as the reward signal. Note that we also consistently plot the logarithmic simple regret in our evaluations for these cases.

C.2.1.6 Neural AF Architecture

We used multi-layer perceptrons with relu-activation functions and four hidden layers with 200 units each to represent the neural AFs.

C.2.1.7 Value Function Network

To reduce the variance of the gradient estimates for PPO, a value function $V_\pi(s_t)$, i.e., an estimator for the expected cumulative reward from state s_t, can be employed [Sch16b]. In this context, the optimization step t and the budget T are particularly informative features, as for a given sampling strategy on a given function class they allow quite reliable predictions of future regrets. Thus, we propose to use a separate neural network to learn a value function of the form $V_\pi(s_t) = V_\pi(t, T)$. We used an MLP with relu-activations and four hidden layers with 200 units each for the value network.

C.2.1.8 Computation Time

For training MetaBO, we employed ten parallel CPU-workers to record the data batches and one GPU to perform the policy updates. Depending on the complexity of the objective function evaluations, training a neural AF for a given function class took between approximately 30 min and 10 h on this moderately complex architecture.