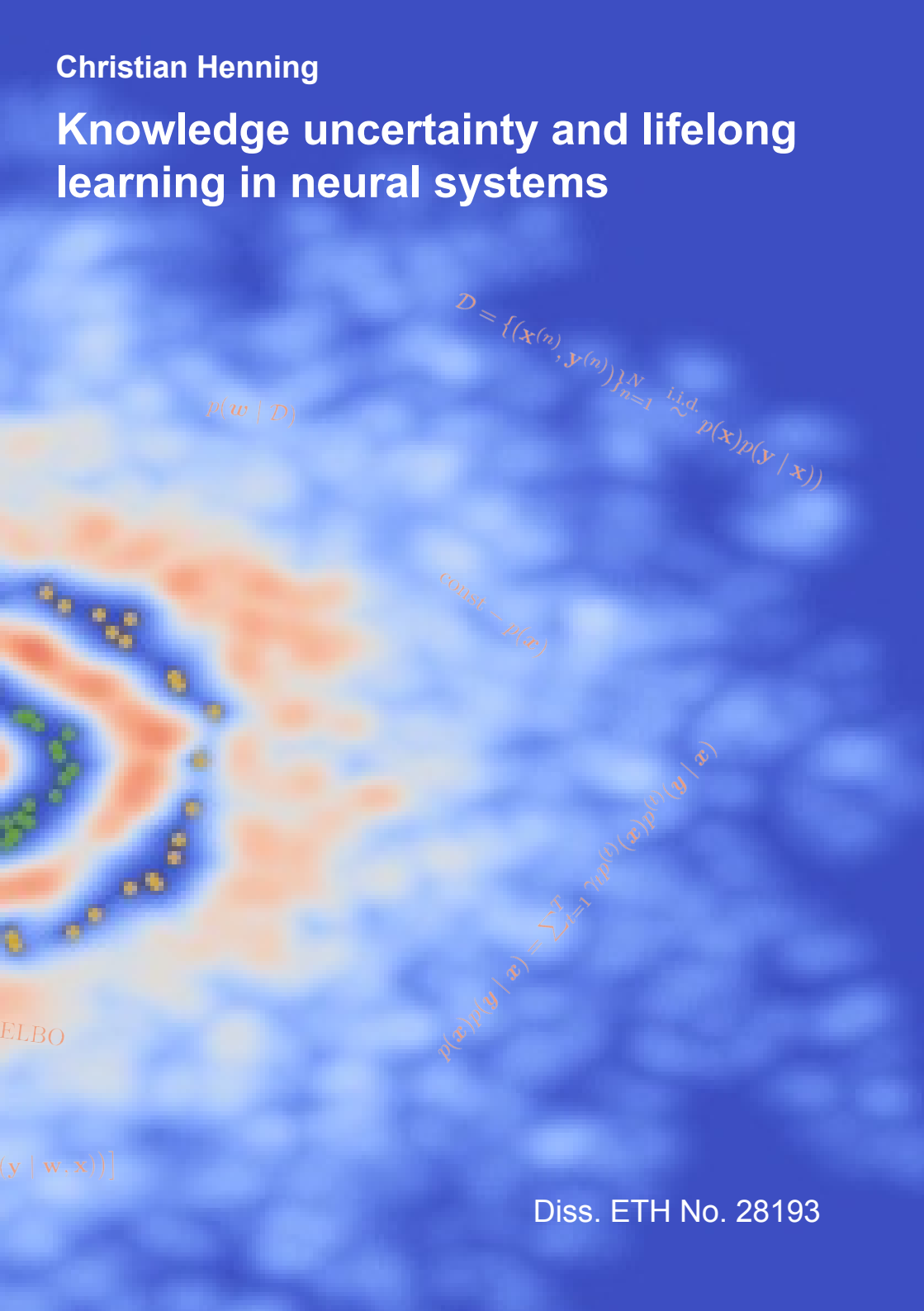
Christian Henning

Knowledge uncertainty and lifelong learning in neural systems



Diss. ETH No. 28193

CHRISTIAN HENNING

KNOWLEDGE UNCERTAINTY AND LIFELONG LEARNING IN NEURAL SYSTEMS

DISS. ETH NO. 28193

KNOWLEDGE UNCERTAINTY AND LIFELONG LEARNING IN NEURAL SYSTEMS

A dissertation submitted to attain the degree of

DOCTOR OF SCIENCES of ETH ZURICH
(Dr. sc. ETH Zurich)

presented by

CHRISTIAN HENNING
MSc., Leibniz Universität Hannover

born on 05 September 1991
citizen of Germany

accepted on the recommendation of

Prof. Dr. Angelika Steger, examiner
Prof. Dr. Joachim Buhmann, co-examiner
Prof. Dr. Jean-Pascal Pfister, co-examiner

2022

Christian Henning: *Knowledge uncertainty and lifelong learning in neural systems,* © 2022

DOI: 10.3929/ethz-b-000523790

dedicated to my niece

Noemi Maria Nikolai

ABSTRACT

Natural intelligence has the ability to continuously learn from its environment, an environment that is constantly changing and thus induces uncertainties that need to be coped with to ensure survival. By contrast, artificial intelligence (AI) commonly learns from data only once during a particular training phase, and rarely explicitly represents or utilizes uncertainties. In this thesis, we contribute towards improving AI in these regards by designing and understanding neural network-based models that learn continually and that explicitly represent several sources of uncertainty, with the ultimate goal of obtaining models that are useful, reliable and practical.

We start by setting this research into a broader context and providing an introduction to the fields of uncertainty estimation and continual learning. This detailed review can constitute an entry point for those interested in familiarizing themselves with these topics. After laying this foundation, we dive into the specific question of how to learn a set of tasks continually and present our approach for solving this problem based on a system of neural networks. More specifically, we train a meta-network to generate task-specific parameters for an inference model and show that, in this setting, forgetting can be prevented using a simple regularization at the meta-level. Due to the existence of task-specific solutions, the problem arises of having to infer the task to which an unseen input belongs. We investigate two major ways for solving this *task-inference* problem: (i) replay-based and (ii) uncertainty-based. While replay-based task-inference exhibits remarkable performance on simple benchmarks, our implementation of this method relies on generative modelling, which becomes disproportionately difficult with increased task complexity. Uncertainty-based task-inference, on the other hand, does not rely on external models and scales more easily to complex scenarios. Because calibrating the uncertainties required for task-inference is difficult, in practice, one often resorts to models that should *know what they don't know*. This can in theory be achieved through a Bayesian treatment of model parameters. However, due to the difficulty in interpreting the prior knowledge given to a neural network-based model, it also becomes difficult to interpret what it is that the model *knows not to know*. This realization has implications beyond continual learning, and more generally affects how current machine learning models handle unseen inputs. We discuss the intricacies associated with choosing prior knowledge

in neural networks and show that common choices often lead to uncertainties that do not intrinsically reflect certain desiderata such as detecting unseen inputs that the model should not generalize to.

Overall, this thesis compactly summarizes and contributes to the advancement of two important topics in nowadays deep learning research, uncertainty estimation and continual learning, while disclosing existing challenges, evaluating novel approaches and identifying promising avenues for future research.

ZUSAMMENFASSUNG

Biologisch-entwickelte Intelligenz besitzt die Fähigkeit kontinuierlich von ihrer Umgebung zu lernen. Diese Umgebung verändert sich ständing, wodurch Unsicherheiten entstehen, die es zu beachten gilt um ein Überleben sicherzustellen. Im Gegensatz dazu lernt künstliche Intelligenz (KI) meist nur während einer gesonderten Trainingsphase von ausgewählten Daten und ist selten entworfen um explizit Unsicherheiten zu repräsentieren oder auszunutzen. Ziel dieser Arbeit ist es, KI in dieser Hinsicht zu verbessern, indem wir neue Methoden mit verbesserter Leistungsfähigkeit präsentieren und zum Verständnis existierender Methoden beitragen.

Zu Beginn dieser Thesis geben wir eine umfangreiche Einführung in die Themen des Schätzens von Unsicherheiten und des lebenslangen Lernens. Diese allgemeine Übersicht kann als Ausgangspunkt dienen, um sich mit den entsprechen Themengebieten vertraut zu machen. Nach der Einführung dieser Grundlagen werden wir uns explizit damit befassen, wie eine Folge unterschiedlicher Aufgaben nacheinander von einer KI gelernt werden kann. Unser Ansatz zu dieser Problemstellung basiert auf einem System von verschachtelten neuronalen Netzwerken oder genauer: wir trainieren ein Meta-Netzwerk um aufgaben-spezifische Parameter für ein Inferenz-Netzwerk zu generieren. In dieser Konfiguration kann das Problem des *Vergessens* auf vereinfachte Weise mittels einer Regularisierung auf der Ebene der aufgaben-spezifischen Parameter angegangen werden. Aufgrund der expliziten Betrachtung von aufgaben-spezifischen Lösungen, muss für jede Eingabe ins Inferenz-Netzwerk zuerst die Identität einer Aufgabe ermittelt werden. Wir untersuchen zwei verschiedene Ansätze um die Aufgabe zu inferieren: (i) mittels expliziten Abrufens von Daten aus vergangenen Aufgaben (*replay*-basiert), oder (ii) anhand von modellierten Unsicherheiten (*uncertainty*-basiert). Für Problemstellungen in denen die Komplexität der Eingabedaten gering ist, funktionieren *replay*-basierte Methoden am besten. Allerdings erfordert unsere Implementation dieser Methoden ein explizites Erlernen eines generativen Modells für Eingabedaten, was mit zunehmender Komplexität der Daten nur schwer möglich ist. Im Gegensatz dazu benötigen *uncertainty*-basierte Methoden kein explizites generatives Modell und können daher auch für komplexere Daten angewendet werden. Da es jedoch kompliziert ist, Unsicherheiten, die für die Inferenz der Aufgaben-Identität von nöten sind, zu kalibrieren, werden

in der Praxis häufig Modelle verwendet, die, einfach ausgedrückt, *sich ihrer Unwissenheit bewusst sind*. Theoretisch können solche Modelle realisiert werden indem Parameter probabilistisch mittels bayesscher Statistik behandelt werden. Jedoch ist es schwierig das benötigte Vorwissen (A-priori Wissen) mit dem das neuronale Netzwerk ausgestattet werden muss zu interpretieren. Daher ist es oft nicht möglich zu verstehen *welcher Unwissenheit sich das Modell bewusst ist*. Diese Erkenntnis hat Konsequenzen über die Frage des lebenslangen Lernens hinaus und beeinflusst, zum Beispiel, wie unbekannte Eingaben von dem System behandelt werden. Wir diskutieren die Besonderheiten, die es zu Beachten gilt, wenn ein neuronales Netzwerk mit Vorwissen ausgestattet wird und zeigen, dass herkömmliche Methoden nicht die gewünschte Fähigkeit besitzen Eingaben zu erkennen, für die sie nicht trainiert wurden.

Kurzgesagt befasst sich diese Arbeit mit zwei wichtigen ungelösten Problemen der künstlichen Intelligenz, dem lebenslangen Lernen und dem Modellieren von Unsicherheiten. Durch die ausführliche Einführung dieser Themen zeigen wir existierende Schwachstellen auf. Mit der anschließenden Präsentation unserer Beiträge zeigen wir wie diese Schwachstellen durch unsere Lösungsvorschläge angegangen werden und heben zusätzlich verbleibende Probleme sowie zukünftige Möglichkeiten hervor.

ACKNOWLEDGEMENTS

I'm deeply indebted to my family for their continuing support and also to Hien Thuong Tran. Thuong was my greatest support throughout my undergrad studies and without her, I probably wouldn't have made it thus far. During my doctoral studies, I had the honor to work with many talented, inspiring and generous people. I'm extremely grateful to these colleagues and friends, and their is no particular order in which to thank them best. Yet, I want to express special gratitude to my colleague Maria R. Cervera. She was the most reliable person I ever worked with and I believe that the challenging projects we have undertaken together would not have been feasible without her hard work, dedication, curiosity and project management. The countless fruitful discussions we had together helped me to grow in my understanding and to discover and crystallize many scientific questions. Furthermore, I would like to extent my deepest gratitude to João Sacramento, whose scientific supervision taught me how to become a scientist and from whose knowledge and experience I could benefit repeatedly when conducting my own research. Furthermore, I would like to thank Benjamin Ehret, Roman Böhringer and Harald Dermutz for their professional and personal support. I am grateful to call them my friends and I am glad we went through this journey together. Another special colleague has been Francesco D'Angelo, who first approached me as student that I soon realized to be particularly hard-working, smart and humble. This is why we quickly became partners and to this date I approach him almost daily to discuss scientific thoughts. I am also grateful to Johannes von Oswald and Alexander Meulemans, which are amazing colleagues that I deeply respect and admire for their competence and personality. I also want to show my respect and gratitude to Prof. Dr. Angelika Steger, who always acted as a role model for scientific integrity and who decided to help me without hesitation when I needed her help. I additionally want to thank Prof. Dr. Benjamin Grewe for his persistent hard work to provide the environment and resources that enabled the research presented in this thesis. I also had great pleasure discussing my research and scientific questions in general with Prof. Dr. Jean-Pascal Pfister. There are many other people from the Grewe and Steger lab, as well as my students, that would deserve all to be listed here, and I am very grateful that I had the chance

to work with them and I very much enjoyed the work environment they provided.

Furthermore, I would like to express my deepest appreciation to my committee consisting of Prof. Dr. Angelika Steger, Prof. Dr. Joachim Buhmann and Prof. Dr. Jean-Pascal Pfister. All of them did an excellent job reviewing my thesis and provided feedback that improved its final version.

Lastly, I would like to especially thank Maria R. Cervera but also Francesco D'Angelo, Benjamin Ehret, João Sacramento and Johannes von Oswald for proofreading and for providing constructive feedback.

CONTENTS

NOTATION

FREQUENTLY USED SYMBOLS

$\boldsymbol{x}$	inputs
$\boldsymbol{y}$	outputs
$\mathcal{D}$	dataset
$\boldsymbol{w}$	parameters – discriminative model
$\boldsymbol{v}$	parameters – generative model
$\boldsymbol{e}$	input embedding – hypernetwork
ψ	parameters – hypernetwork
ϕ	parameters – approximate posterior

CONVENTIONS

x, y	scalars
$\boldsymbol{x}, \boldsymbol{y}$	bold letter – vector
X, Y	capital letters – matrix
$\mathcal{X}, \mathcal{Y}$	calligraphy capital letters – set
$f(\cdot), h(\cdot)$	parametrized functions (e. g., neural networks)
$p(\cdot), q(\cdot)$	probability density or mass function

1

INTRODUCTION

It would be better for us to have some doubts in an honest pursuit of truth, than it would be for us to be certain about something that was not true.
— Daniel Wallace

Can we formulate and pursue scientific questions that will allow us to unlock some of the mysteries of the brain such that we can translate them into technology? This is a fundamental question for every artificial intelligence researcher whose intrinsic motivation is to understand the essence of human intelligence rather than to just build useful systems that automate processes. Unfortunately, the brain is overwhelmingly complex and no agreed-upon recipe exists on how to study it best.

The initial approach for learning from the brain that I pursued in my PhD was to (i) come up with a hypothesis, (ii) develop an experimental paradigm for testing this hypothesis in an animal model (mice, in my case), and (iii) use this hypothesis to develop theoretical algorithms that ultimately improve performance on target applications. Yet, this complex, interdisciplinary pipeline completely relies on the usefulness of the initially declared hypothesis. If the motivation is to understand some of the core elements that intuitively make up human intelligence, then defining a hypothesis is like providing an answer to a question that has not been sufficiently studied yet. For instance, we will outline in Chapter 4 that the answer to the question "How to continually learn?" may have very different answers based on what is understood when referring to "continual learning". This realization, that forming a hypothesis first requires a better understanding of what is the essence of intelligence, made me change my approach to science. Thus, rather than hypothesizing how the brain implements a vague property of what we consider intelligence, we can try to formalize these properties and reason about possible ways in which they could be implemented independent of biological constraints. This paradigm describes what machine learning research is to me: a community effort that tries to iteratively define more and more precisely what useful properties of intelligence are, while coming up with many alternative algorithms of how these properties can be realized, where better algorithms crystallize

over time due to the competitive nature of the field. This reasoning defines the spirit underlying this thesis, namely that only when a computational feature of intelligence is sufficiently well defined, and multiple alternative and well-performing algorithms for realizing it are known, can a useful hypothesis be constructed on how this feature might emerge in the brain.

The computational features this thesis is concerned with are *continual* (or *lifelong*) *learning* and *uncertainty estimation*. Those features would augment a model with the desirable capabilities of updating the model's knowledge upon receiving new evidence and to be aware of the limitations of its own knowledge when making predictions. Over the course of this thesis, we will precisely define those intuitive notions and investigate ways of realizing those features in artificial systems.

Before we dive into the formal treatment of these problems, we conclude this introduction by motivating the research topics uncertainty estimation (Sec. 1.1) and continual learning (Sec. 1.2). The rest of this thesis is structured as follows. In Chapter 2, we give a brief presentation of the general background necessary to follow this thesis, before an in-depth discussion on the main topics of uncertainty estimation (Chapter 3) and continual learning (Chapter 4) is provided. The remaining chapters will present some of the projects that I was working on during my PhD. In Chapter 5, we discuss how continual learning can be addressed at a meta-level. A probabilistic extension of this framework is then introduced in Chapter 6. In Chapter 7, we discuss difficulties that arise when employing Bayesian statistics to neural networks in order to augment a model with the ability to *know what it does not know*. Finally, we conclude this thesis with a discussion and outlook in Chapter 8.

1.1 UNCERTAINTY ESTIMATION AS A FEATURE OF INTELLIGENCE

Uncertainty arises naturally in our world due to imperfect perception and imperfect knowledge, but also due to the element of choice. Intelligence is goal-oriented, ultimately driven to ensure survival, and there commonly exist multiple ways that can be chosen to achieve the same goal (cf. Fig. 3.1). Estimating such uncertainties properly is undoubtedly useful for decision making [e. g., 1, 2], but how to do so in artificial systems remains a practical challenge (cf. Chapters 3 and 7). The type of uncertainty that will be discussed most in this thesis is the one arising due to imperfect knowledge (cf. Sec. 3.1). As there is an intrinsic need to reduce uncertainty, information-gathering or even curiosity can be seen as uncertainty-driven means to

increase one's knowledge, highlighting one of many useful aspects of maintaining such uncertainty.

Uncertainties can be expressed via probabilities and, as knowledge is subjective, also these probabilities will ultimately be subjective [3]. Bayesian statistics allows us to formally represent such subjective beliefs in terms of probability theory [4] and thus provides us with a principled tool to equip models with the ability to *know what they don't know* (cf. Sec. 3.2 and Sec. 3.3). While all our elaborations are rooted in taking a Bayesian perspective, it remains an ongoing debate whether natural intelligence approximates Bayesian statistics for capturing knowledge uncertainties (e. g., see the work by Bowers & Davis [5] or by Marcus & Davis [6] and Goodman *et al.* [7]). But apart from the discussion whether knowledge uncertainties are represented in a principled or heuristical manner, it is, to the best of our knowledge, uncontroversial to assume these uncertainties are used in decision-making (e. g., see work by Jegminat *et al.* [2] for recent evidence).[1]

In our work, we focus on a class of parametric models, called *neural networks* (Sec. 2.2). In Chapter 3, we will see how the lack of knowledge can, at least partially, be captured within predictive uncertainties by an explicit treatment of parameter uncertainty. In biological neural systems, uncertainty over synaptic weights might play a similar role, e. g., Ref. [10]. Ignoring parameter uncertainty when estimating the parameters of a model can have detrimental consequences in all fields of science (e. g., by confidently overestimating the amount of intelligent life in the universe [11]). Yet, most modern deep learning approaches use a single parameter estimate for making predictions. One reason for this is surely computational convenience. Another reason, however, is the questionable *usefulness* of uncertainty estimates stemming from a naive application of Bayesian statistics.

There are two main challenges intrinsic to Bayesian deep learning. The first is the need to resort to approximate inference (cf. Sec. 3.5), which can yield vastly different uncertainty estimates in practice (cf. Chapter 6). Approximate inference also makes it additionally challenging to assess the validity of the chosen prior knowledge, which is the second main challenge. This is important since knowing what we don't know ultimately depends on both, the observed data and the knowledge that was available a priori (cf. Sec. 3.3). While natural intelligence might have evolved to possess useful prior knowledge, we will see in Chapter 7 how difficult it currently is to

1 Also in deep learning, knowledge uncertainties can be represented heuristically, e.g., through *deep ensembles* [8], which have no known mathematically-derived Bayesian interpretation in their original formulation [9].

encode meaningful prior knowledge into neural networks. We will discuss these challenges in Chapters 3 and 7, but also emphasize that scientific progress can overcome them to increase the trustworthiness associated with the "black-box" of deep learning (cf. Sec. 3.6).

1.2 ARTIFICIAL SYSTEMS THAT CAN CONTINUALLY LEARN

The second aspect of intelligence on which this thesis focuses is continual learning. Commonly, a neural network's parameters are trained and then frozen before deployment. This is in stark contrast to how biological agents learn, which can lifelong adapt to their changing environment (even though there might be periods of increased plasticity [12]). It is natural and economical to ask how artificial systems can be equipped with the same trait, as neural network training is costly and improvements that could be made by incorporating new data should ideally not require a complete retraining.

It is long known that a naive incorporation of new data leads to interference with existing knowledge [13, 14]. These observations have famously sparked interest in interpreting the role of the hippocampus for memory formation in the brain [15, 16] as a means to allow continual learning without interference (cf. Sec. 4.1.1). Complementary to this, there is increasing evidence that the brain uses specialized mechanisms to prevent forgetting [17], and some of these mechanisms can also be considered as the inspiration behind continual learning algorithms for neural networks [18].

In addition to that, Bayesian statistics, which we discussed in the previous section as a means to capture knowledge uncertainty, prescribes how to optimally incorporate new data (cf. Sec. 4.1.2). Even though this optimal approach lacks strong evidence of practical feasibility in parametric models that require approximate inference (cf. Sec. 4.1.2.1), Bayesian statistics can also inspire other approaches to continual learning (e. g., our method proposed in Chapter 6), that yield state-of-the-art performances.

We will provide a detailed introduction into this topic in Chapter 4, including our own nomenclature of current methods and an alternative to the common perspective on the problem formulation. We will then present our own methods for continual learning in Chapters 5 and 6.

2

BACKGROUND

In this chapter, we introduce the statistical framework for learning that we consider in this thesis. After having built an understanding of what it means to learn from data, we will present neural networks which are the predominant machine learning model studied in the following chapters.

2.1 SUPERVISED LEARNING AND THE DATA-GENERATING PROCESS

In this section, we define the problem of "learning" and reduce it to the problem of "function approximation". For learning to be targeted, we first need to introduce the concept of a data-generating process that will provide us with observations. Those observations will guide our learning algorithm to infer certain properties of the data-generating process. Overall, the goal of learning from observations, i.e., to "train" a system, is that we can mimic useful aspects of the data-generating process. For instance, assume our goal is to learn how to steer a car [e. g., 19], where observations are tuples $(\boldsymbol{x}_{\text{image}}, y_{\text{angle}})$ of simultaneous recordings of an image $\boldsymbol{x}_{\text{image}}$ from the car's front camera and a steering angle y_{angle} taken from the car's bus system. Assume these observations are generated by recording a human's driving behavior. The goal of learning would now be to mimic the human driving and thus to train a system that upon receiving an "unseen" front camera image $\boldsymbol{x}_{\text{image}}$ (assuming this is the same input that is perceived by the human driver) can predict the steering angle that would be chosen by the human driver. The word "unseen" is the crucial factor that distinguishes learning from memorization. A car's front camera will likely never record the same image twice, which makes it necessary to deduce a rule from the given observations that is general enough to capture the driver's behavior. We now formally introduce those basic concepts as necessary in the context of this thesis. However, this section does not provide a complete coverage of machine learning paradigms nor does it serve as a proper introduction into statistical learning theory. Please refer to, e.g., the work by Hastie, Tibshirani & Friedman [20] for more details.

2.1.1 *Data generation from a probabilistic perspective*

In most cases, we are interested in learning a relationship between inputs $\boldsymbol{x}$ and outputs $\boldsymbol{y}$ as in the steering angle prediction example above. If not mentioned otherwise, we assume that inputs are real-valued $\boldsymbol{x} \in \mathcal{X} \subseteq \mathbb{R}^{n_{\text{in}}}$. For outputs, we distinguish two different learning problems based on whether they are discrete or continuous.

Definition 1 *Learning to predict outputs $\boldsymbol{y} \in \mathcal{Y}$ which are continuous $\mathcal{Y} \subseteq \mathbb{R}^{n_{out}}$ is called a **regression** problem. If outputs are categorical $\mathcal{Y} = \{1, \dots, K\}$, the problem is called a K-way **classification**.*

We take a probabilistic view on the world and assume that the generation of data can always be explained by the use of probability distributions. Specifically, we assume an input distribution $P(X = \boldsymbol{x})$ from which input observations are sampled from. Sometimes, we also want to approximate this distribution explicitly in terms of its probability density function $p(\boldsymbol{x})$. Note, that in this case we additionally need to assume that the input distribution has a tractable density function. For the sake of readability, we will generally not distinguish in our notation between a distribution and its corresponding probability density function.

To define a proper joint that observations can be sampled from, we also need a conditional $p(\boldsymbol{y} \mid \boldsymbol{x})$. Later, we will see that learning in the context of this thesis often refers to approximating the conditional $p(\boldsymbol{y} \mid \boldsymbol{x})$, which is why we always assume that the induced probability mass (or density) function is well defined within the support of $p(\boldsymbol{x})$.

Definition 2 *The data-generating process is defined via a joint $p(\boldsymbol{x})p(\boldsymbol{y} \mid \boldsymbol{x})$. This joint is unknown to the learning algorithm and is generally referred to as ground-truth. We assume access to observations via a **dataset***

$$\mathcal{D} \coloneqq \{(\boldsymbol{x}^{(n)}, \boldsymbol{y}^{(n)})\}_{n=1}^{N} \overset{i.i.d.}{\sim} p(\boldsymbol{x})p(\boldsymbol{y} \mid \boldsymbol{x}) \quad . \tag{2.1}$$

*This setting, where we have access to ground-truth tuples of inputs $\boldsymbol{x}$ and outputs $\boldsymbol{y}$ defines a machine learning paradigm called **supervised learning**.*

In our initial example, observations are recorded from a human driving experience, and thus consecutive observations are autocorrelated. From this perspective, the assumption in Def. 2, that a dataset represents an i.i.d. sample of the data-generating process might seem unrealistic. Indeed, this assumption represents a major hurdle in many practical learning scenarios and often requires careful data curation. When discussing continual

learning in Chapter 4 we will encounter scenarios where this assumption is violated such that learning algorithms need to be developed that can operate in a less restrictive setting. For now, however, we assume this assumption to be satisfied.

2.1.2 *Learning as conditional density estimation*

In the previous subsection, we have seen that observations stem from a joint $p(\boldsymbol{x})p(\boldsymbol{y} \mid \boldsymbol{x})$ which represents the data-generating process. We introduced the goal of learning from these observations as to being able to mimic useful aspects of this data-generating process, a notion that we can now formalize.

Definition 3 *Using $\mathcal{D}$ to capture $p(\boldsymbol{x})$ is called **generative modelling**, and using $\mathcal{D}$ to capture $p(\boldsymbol{y} \mid \boldsymbol{x})$ is called **discriminative modelling**.*

The above definition is purposely vague in using the term "capture". It might be that the goal of learning is to capture a statistic from the underlying distribution such as the mean of $p(\boldsymbol{y} \mid \boldsymbol{x})$, or to attain the ability of generating new sample points that appear to be sampled from $p(\boldsymbol{x})$. In the course of this thesis, we will encounter many context-dependent desiderata that will define what aspect of $p(\boldsymbol{x})p(\boldsymbol{y} \mid \boldsymbol{x})$ we would like to capture. However, we will ubiquitously encounter the concept of **maximum likelihood** training, especially in the context of learning a discriminative model. To introduce this concept, we first have to explain the meaning of a likelihood function and its corresponding hypothesis class.

Assume our goal is to approximate $p(\boldsymbol{x})$ (or $p(\boldsymbol{y} \mid \boldsymbol{x})$) using a family of distributions $p(\boldsymbol{x} \mid \boldsymbol{v})$ (or $p(\boldsymbol{y} \mid \boldsymbol{w}, \boldsymbol{x})$) parametrized by $\boldsymbol{v} \in \mathcal{V}$ (or $\boldsymbol{w} \in \mathcal{W}$).[1] We purposely choose a notation where we treat the distributional parameters as random variables that we condition on. The reason for choosing such notation will become clear in Sec. 3.2 when we apply Bayesian statistics to this framework. For now, one can consider parameters simply as a chosen constant, e.g., $p(\boldsymbol{x}; \boldsymbol{v})$ (or $p(\boldsymbol{y} \mid \boldsymbol{x}; \boldsymbol{w})$).

The chosen family of distributions (e.g., $p(\boldsymbol{x} \mid \boldsymbol{v})$) is what we refer to as *model*. An instance of this model (in this case, an explicit density (or mass) function) defined by an individual parametrization is called *hypothesis*. The space of all hypotheses induced by the parameter space $\mathcal{V}$ (or $\mathcal{W}$) is called *hypothesis space*. If there is no hypothesis in this space that faithfully

1 Note, that we overload notation, and $p(\cdot)$ or $p(\cdot \mid \cdot)$ may refer to different density functions based on the variable names used as arguments.

resembles the ground-truth, we say the model is misspecified. We briefly discuss model misspecification in Sec. 2.1.3.

To choose a hypothesis from this hypothesis space that best resembles the data, we need a fitness or cost function that allows us to compare hypotheses. Specifically, for every hypothesis (indexed by a specific parametrization), we want a scalar function $\mathcal{L}^{\text{gen}} : \mathcal{V} \to \mathbb{R}$ (or $\mathcal{L}^{\text{dis}} : \mathcal{W} \to \mathbb{R}$) that achieves its lowest value for the hypothesis that is considered best under some chosen criterion. Let's look at an example for such criterion.

Example 1 – *Maximum likelihood training of a generative model.* *Let's assume our goal is to find the parameters $\boldsymbol{v}$ for our model $p(\boldsymbol{x} \mid \boldsymbol{v})$ that best approximate the ground-truth $p(\boldsymbol{x})$ using the dataset $\mathcal{D}$. If we define an approximation error in terms of the forward Kullback-Leibler (KL) divergence, we get the following expression:*

$$KL\left(p(\boldsymbol{x}); p(\boldsymbol{x} \mid \boldsymbol{v})\right) = -\mathcal{H}\left\{p(\boldsymbol{x})\right\} - \mathbb{E}_{p(\boldsymbol{x})}\left[\log p(\boldsymbol{x} \mid \boldsymbol{v})\right] \tag{2.2}$$

Since the entropy $\mathcal{H}\left\{p(\boldsymbol{x})\right\}$ is constant, we can minimize the KL by minimizing the cross-entropy, which can be approximated via Monte-Carlo using the i.i.d. dataset $\mathcal{D}$:

$$-\mathbb{E}_{p(\boldsymbol{x})}\left[\log p(\boldsymbol{x} \mid \boldsymbol{v})\right] \approx -\frac{1}{N}\sum_{n=1}^{N} \log p(\boldsymbol{x}^{(n)} \mid \boldsymbol{v}) \tag{2.3}$$

This allows us to write down a tractable optimization objective, assuming density evaluation under the chosen model $p(\boldsymbol{x} \mid \boldsymbol{v})$ is tractable:

$$\arg\min_{\boldsymbol{v}} -\sum_{n=1}^{N} \log p(\boldsymbol{x}^{(n)} \mid \boldsymbol{v}) = \arg\max_{\boldsymbol{v}} \prod_{n=1}^{N} p(\boldsymbol{x}^{(n)} \mid \boldsymbol{v}) \tag{2.4}$$

The RHS of Eq. 2.4 can be interpreted as maximizing the likelihood of the observed data. For this reason, the function $\mathcal{L}^{gen}(\boldsymbol{v}) \coloneqq -\sum_{n=1}^{N} \log p(\boldsymbol{x}^{(n)} \mid \boldsymbol{v})$ is called the negative log-likelihood loss.

Now, the focus of this work lies on discriminative modelling. Luckily, an analogous maximum likelihood criterion as in Example 1 can be derived for learning the parameters $\boldsymbol{w}$ of $p(\boldsymbol{y} \mid \boldsymbol{w}, \boldsymbol{x})$

$$\mathbb{E}_{p(\boldsymbol{x})}\left[\mathrm{KL}\left(p(\boldsymbol{y} \mid \boldsymbol{x}); p(\boldsymbol{y} \mid \boldsymbol{w}, \boldsymbol{x})\right)\right] = \text{const.} - \mathbb{E}_{p(\boldsymbol{x})p(\boldsymbol{y}|\boldsymbol{x})}\left[\log p(\boldsymbol{y} \mid \boldsymbol{w}, \boldsymbol{x})\right] \tag{2.5}$$

Again, we can use the dataset $\mathcal{D}$ to obtain a Monte-Carlo estimate of Eq. 2.5, which for discriminative models leads to the following **negative log-likelihood** (NLL) loss:

$$\mathcal{L}^{\text{dis}}(\boldsymbol{w}) \coloneqq -\sum_{n=1}^{N} \log p(\boldsymbol{y}^{(n)} \mid \boldsymbol{w}, \boldsymbol{x}^{(n)}) \tag{2.6}$$

This loss function is commonly used in supervised learning, even though it might be known under different names depending on the model choice $p(\boldsymbol{y} \mid \boldsymbol{w}, \boldsymbol{x})$ as we will outline below. Note, that minimizing this loss corresponds to performing maximum likelihood estimation (MLE) $\boldsymbol{w}^{\text{MLE}} = \arg\max_{\boldsymbol{w}} -\mathcal{L}^{\text{dis}}(\boldsymbol{w})$.

Viewing the objective in Eq. 2.6 as an approximation to minimizing the expected KL-divergence in Eq. 2.5 gives rise to the interpretation of (supervised) learning just being conditional density estimation. When looking at how we define a model $p(\boldsymbol{y} \mid \boldsymbol{w}, \boldsymbol{x})$ in practice, we can further reduce the notion of (supervised) learning as function approximation as illustrated in the following examples.

Example 2 – *Mean-squared error loss.* *Let's consider a univariate regression problem, i.e., $y \in \mathbb{R}$. Furthermore, we assume that the model $p(y \mid \boldsymbol{w}, \boldsymbol{x})$ represents the family of Gaussian distributions with $\boldsymbol{x}$-dependent mean and fixed variance: $p(y \mid \boldsymbol{w}, \boldsymbol{x}) = \mathcal{N}\left(y; f(\boldsymbol{x}; \boldsymbol{w}), \sigma^2\right)$. Thus, the goal of learning is to estimate the parameters $\boldsymbol{w}$ of a function $f : \mathcal{X} \times \mathcal{W} \to \mathbb{R}$ using the NLL loss*

$$-\frac{1}{N}\sum_{n=1}^{N} \log p(y^{(n)} \mid \boldsymbol{w}, \boldsymbol{x}^{(n)}) = \textit{const.} + \frac{1}{N}\sum_{n=1}^{N} \frac{\left(y^{(n)} - f(\boldsymbol{x}^{(n)}; \boldsymbol{w})\right)^2}{2\sigma^2} \tag{2.7}$$

Given that we assume the variance σ^2 to be fixed, we can connect maximum likelihood estimation with the commonly used mean-squared error loss:

$$\boldsymbol{w}^{MLE} = \arg\min_{\boldsymbol{w}} \frac{1}{N}\sum_{n=1}^{N} \left(y^{(n)} - f(\boldsymbol{x}^{(n)}; \boldsymbol{w})\right)^2 \tag{2.8}$$

Example 3 – *Cross-entropy loss.* *Consider a K-way classification problem such that $p(y \mid \boldsymbol{x})$ is a categorical distribution. We represent the model's probability mass function for a given $\boldsymbol{x}$ and $\boldsymbol{w}$ via a vector $\boldsymbol{p}(\boldsymbol{x}; \boldsymbol{w})$ with entries $p_k(\boldsymbol{x}; \boldsymbol{w}) \equiv p(k \mid \boldsymbol{w}, \boldsymbol{x})$. This vector can be obtained via normalization of the outputs of a real-valued function $f : \mathcal{X} \times \mathcal{W} \to \mathbb{R}^K$, for instance, by using a softmax function*

$p_k(\boldsymbol{x}; \boldsymbol{w}) = \exp\left(f(\boldsymbol{x}; \boldsymbol{w})_k\right) / \sum_{k'} \exp\left(f(\boldsymbol{x}; \boldsymbol{w})_{k'}\right)$. *Hence, we again aim to learn the parameters of a function by minimizing the NLL loss:*

$$-\frac{1}{N}\sum_{n=1}^{N} \log p(y^{(n)} \mid \boldsymbol{w}, \boldsymbol{x}^{(n)}) = -\frac{1}{N}\sum_{n=1}^{N} \log p_{y^{(n)}}(\boldsymbol{x}^{(n)}; \boldsymbol{w}) \tag{2.9}$$

As the NLL loss is just a numerical approximation to the expected cross entropy $\mathbb{E}_{p(\boldsymbol{x})}\left[\mathcal{H}\left\{p(y \mid \boldsymbol{x}), p(y \mid \boldsymbol{w}, \boldsymbol{x})\right\}\right]$*, this loss is commonly referred to as cross-entropy loss in the context of classification tasks. When using a softmax for normalization, the loss can be written as:*

$$\boldsymbol{w}^{MLE} = \arg\min_{\boldsymbol{w}} -\frac{1}{N}\sum_{n=1}^{N}\left(f(\boldsymbol{x}^{(n)}; \boldsymbol{w})_{y_n} - \log \sum_k \exp\left(f(\boldsymbol{x}^{(n)}; \boldsymbol{w})_k\right)\right) \tag{2.10}$$

The Examples 2 and 3 provide the base for almost all discriminative learning scenarios that will be encountered in this thesis. In Sec. 2.2 we will define how to choose a class of parametrized functions and how learning, i.e., loss optimization, can take place in practice.

Note, that even (or especially) if optimization is perfect, the estimate $\boldsymbol{w}^{\text{MLE}}$ obtained from the statistical sample $\mathcal{D}$ is not necessarily resembling the ground-truth due to an approximation error for finite-sample sizes when moving from Eq. 2.5 to Eq. 2.6. Specifically, the solutions to the following two optimization problems might be different:

$$\boldsymbol{w}^{(*)} := \arg\min_{\boldsymbol{w}} \mathbb{E}_{p(\boldsymbol{x})}\left[\mathrm{KL}\left(p(\boldsymbol{y} \mid \boldsymbol{x}); p(\boldsymbol{y} \mid \boldsymbol{w}, \boldsymbol{x})\right)\right] \qquad \text{and} \tag{2.11}$$

$$\boldsymbol{w}^{(MLE)} := \arg\min_{\boldsymbol{w}} -\sum_{n=1}^{N} \log p(\boldsymbol{y}^{(n)} \mid \boldsymbol{w}, \boldsymbol{x}^{(n)}) \tag{2.12}$$

Ideally, we want the two hypotheses $p(\boldsymbol{y} \mid \boldsymbol{w}^{(*)}, \boldsymbol{x})$ and $p(\boldsymbol{y} \mid \boldsymbol{w}^{(MLE)}, \boldsymbol{x})$ to be similar. Whether this goal, which is commonly referred to as good **generalization**, can be achieved depends on a number of factors such as (i) the quality and size of the dataset $\mathcal{D}$, (ii) the complexity of the hypothesis space and (iii) the optimization criterion and algorithm. In practice, one can foster generalization through the use of an additional validation set $\mathcal{D}^{\text{val}} \overset{\text{i.i.d.}}{\sim} p(\boldsymbol{x})p(\boldsymbol{y} \mid \boldsymbol{x})$, and test for generalization capabilities using an additional test set $\mathcal{D}^{\text{test}} \overset{\text{i.i.d.}}{\sim} p(\boldsymbol{x})p(\boldsymbol{y} \mid \boldsymbol{x})$ that has not influenced the process

of choosing a hypothesis. Please refer to the work by Mohri, Rostamizadeh & Talwalkar [21] for a more detailed introduction into this topic.

Once an estimate $\hat{\boldsymbol{w}}$ (e.g., $\hat{\boldsymbol{w}} \equiv \boldsymbol{w}^{(MLE)}$) of the parameters is obtained, we can use the hypothesis $p(\boldsymbol{y} \mid \hat{\boldsymbol{w}}, \boldsymbol{x}^*)$ to make predictions on unseen inputs $\boldsymbol{x}^*$. An obvious choice for making predictions would be to sample from the hypothesis $\boldsymbol{y}^* \sim p(\boldsymbol{y} \mid \hat{\boldsymbol{w}}, \boldsymbol{x}^*)$. In practice, the most likely prediction $\boldsymbol{y}^* \equiv \arg\max_{\boldsymbol{y}} p(\boldsymbol{y} \mid \hat{\boldsymbol{w}}, \boldsymbol{x}^*)$ is typically chosen. In Example 2, $\boldsymbol{y}^*$ would thus correspond to the mean $f(\boldsymbol{x}^*; \hat{\boldsymbol{w}})$.

2.1.3 *On model misspecification and axiomatic beliefs*

To conclude this introductory section to machine learning, we discuss some of the practical pitfalls that are intrinsic to the described framework for supervised learning. The previous sections stated a lot of assumptions to arrive at a system that allows making predictions on unseen input data by mimicking some unknown data-generative process. The consequences of some of these assumptions are well studied and anticipated. Other assumptions, like the i.i.d.-ness of the observed dataset, are actively being studied (cf. Chapter. 4). The core belief, however, is the declared existence of a data-generating process $p(\boldsymbol{x})p(\boldsymbol{y} \mid \boldsymbol{x})$ (Def. 2). Apart from the philosophical question whether the world can be described in terms of probability theory, this assumption might be trivially violated as a result of how the data was collected. For instance, consider the introductory example of mimicking a human's steering response upon receiving the car's front camera image. The human driver obviously does not receive the actual front camera image as input, and moreover might use auxiliary information (e.g., coming from the side mirrors) to predict a steering angle. Thus, the input $\boldsymbol{x}$ might not contain

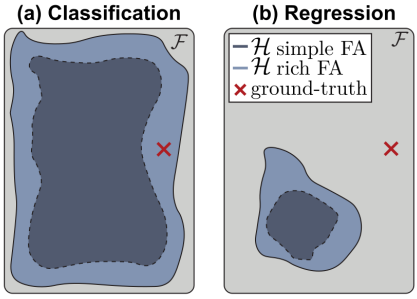


FIGURE 2.1: Figure adapted from [1], showing the hypothesis space $\mathcal{H}$ as induced by a simple or expressive function approximator (FA). For categorical distributions (a), the richness of $\mathcal{H}$ depends mostly on function approximation capabilities. For continuous distributions (b), it also depends on the chosen distributional family.

enough information to reliably predict $\boldsymbol{y}$, a limitation intrinsic to the data that cannot be corrected for, independent of any modelling assumption that determines the assumed relation between these two variables.[2]

Nevertheless, in this thesis, we will never question the existence of some ground-truth $p(\boldsymbol{y} \mid \boldsymbol{x})$ that can be learned from the observed data. Furthermore, in the following chapters, and except mentioned otherwise, we assume the model to be well chosen, meaning that there is a hypothesis in our hypothesis space that matches the ground-truth, i.e., there exists a $\boldsymbol{w}^{(*)} \in \mathcal{W}$ such that $\mathbb{E}_{p(\boldsymbol{x})}\big[\mathrm{KL}\big(p(\boldsymbol{y} \mid \boldsymbol{x}); p(\boldsymbol{y} \mid \boldsymbol{w}^{(*)}, \boldsymbol{x})\big)\big] = 0$ (see our remarks on *model uncertainty* in Sec. 3.1). We study the consequences of model misspecification in the context of neural networks (which are introduced in Sec. 2.2) in [1]. To provide the reader with an intuition for this topic, we quickly discuss how modelling assumptions determine the hypothesis space and provide an example that illustrates how predictions can be agnostic to model misspecification.

Fig. 2.1 sketches the space $\mathcal{F}$ of all imaginable hypotheses (e.g., all possible conditional density functions). We saw in Example 3 that all categorical distributions can be realized (to arbitrary precision) by normalizing a real-valued vector, which can be the output of an x-dependent function $f(\boldsymbol{x}; \boldsymbol{w})$. Hence, to what extent $\mathcal{F}$ is covered by our hypothesis space $\mathcal{H}$ with such model only depends on the richness of the function space from which $f(\boldsymbol{x}; \boldsymbol{w})$ can be selected (Fig. 2.1a), noting that, for instance, neural networks are universal function approximators in the non-parametric limit [23, 24]. This is fundamentally different from the model described in Example 2, where only the mean of a Gaussian distribution is learnt via an x-dependent function $f(\boldsymbol{x}; \boldsymbol{w})$. Hence, no matter how rich the function space is from which we can draw, $p(\boldsymbol{y} \mid \boldsymbol{w}, \boldsymbol{x})$ will always be Gaussian. This is a strong assumption when modelling real world data, and can have detrimental effects on uncertainty estimation [1]. However, if we understand the consequences of such wrong modelling assumptions, we might prefer working with a misspecified model rather than searching an ever bigger hypothesis space, as illustrated by the following example.

Example 4 – *Model misspecification in regression.* *Let's again consider the model from Example 2, $p(y \mid \boldsymbol{w}, \boldsymbol{x}) = \mathcal{N}\left(y; f(\boldsymbol{x}; \boldsymbol{w}), \sigma^2\right)$, while using the following optimization problem to mimic the data-generative process [e.g., 25]:*

2 The driver's state (e.g., tiredness) also influences the steering behavior. However, this affects the stationarity assumption on $p(\boldsymbol{y} \mid \boldsymbol{x})$, which is a modelling choice that in principle can be taken into consideration [22]. Another example is sensor degradation, which might cause $p(\boldsymbol{x})$ to be non-stationary.

$$\boldsymbol{w}^{(*)} \coloneqq \arg\min_{\boldsymbol{w}} \mathbb{E}_{p(\boldsymbol{x})} \left[KL\left(p(y \mid \boldsymbol{x}); \mathcal{N}\left(y; f(\boldsymbol{x}; \boldsymbol{w}), \sigma^2 \right) \right) \right] \tag{2.13}$$

$$= \arg\min_{\boldsymbol{w}} \mathbb{E}_{p(\boldsymbol{x})p(y|\boldsymbol{x})} \left[const. + \frac{1}{2\sigma^2} \left(y - f(\boldsymbol{x}; \boldsymbol{w}) \right)^2 \right] \tag{2.14}$$

$$= \arg\min_{\boldsymbol{w}} \mathbb{E}_{p(\boldsymbol{x})p(y|\boldsymbol{x})} \left[y^2 - 2y f(\boldsymbol{x}; \boldsymbol{w}) + f(\boldsymbol{x}; \boldsymbol{w})^2 \right] \tag{2.15}$$

$$= \arg\min_{\boldsymbol{w}} \mathbb{E}_{p(\boldsymbol{x})} \left[\underbrace{Var\left\{ p(y \mid \boldsymbol{x}) \right\}}_{const.} + \left(\mathbb{E}_{p(y|\boldsymbol{x})}[y] - f(\boldsymbol{x}; \boldsymbol{w}) \right)^2 \right] \tag{2.16}$$

The above calculation shows that the optimum is uniquely attained if the learned mean $f(\boldsymbol{x}; \boldsymbol{w})$ matches the ground-truth's mean $\mathbb{E}_{p(y|\boldsymbol{x})}[y]$ almost everywhere. In Sec. 2.1.1, we discussed taking the mean $f(\boldsymbol{x}; \boldsymbol{w})$ as our prediction, since it's the most likely realization if the Gaussian model assumption is correct. However, even if this model assumption is knowingly incorrect, using an estimate of the ground-truth's mean $\mathbb{E}_{p(y|\boldsymbol{x})}[y]$ as prediction might be reasonable for certain applications. Thus, following the philosophy of George Box, this example illustrates that even wrong models may be useful.

2.2 NEURAL NETWORKS AND GRADIENT-BASED OPTIMIZATION

The previous section presented the learning framework considered in this thesis. We showed that (supervised) learning in this framework can essentially be understood as function approximation. Specifically, the goal is to estimate the parameters $\boldsymbol{w}$ of a function $f(\boldsymbol{x}; \boldsymbol{w})$ from a dataset $\mathcal{D}$, such that $f(\boldsymbol{x}; \boldsymbol{w})$ induces a predictive distribution $p(\boldsymbol{y} \mid \boldsymbol{w}, \boldsymbol{x})$ that resembles the data generating process $p(\boldsymbol{y} \mid \boldsymbol{x})$ for $\boldsymbol{x} \sim p(\boldsymbol{x})$ (cf. Def. 2).

However, we haven't discussed yet how a function approximator can be designed and how its parameters can be estimated from data. This will be the goal of this section, focusing on a specific class of function approximators, called **neural networks**. These are hierarchical models which due to modern compute capabilities can be rather deep, which has proven practically relevant and has made deep neural networks in combination with efficient gradient-based optimization the main driver behind the advent of what is called **deep learning**. See the work by Goodfellow, Bengio & Courville [26] for a thorough introduction into the topics of this section.

As the name indicates, (artificial) neural networks are machine learning models which are loosely inspired by the brain, and were first investigated by McCulloch & Pitts [27]. Most modern neural networks use as elemen-

tary computational unit an abstract model of a biological neuron, called **perceptron** [28]. The transfer function of this model can be described as $y = \Theta\big[b + \sum_{i=1}^{\dim(\boldsymbol{x})} w_i x_i\big]$, where $\Theta[\cdot]$ denotes the Heaviside step function and $\{b, w_1, \ldots, w_{\dim(\boldsymbol{x})}\}$ are learnable parameters. The biological analogy goes as follows, w_i represent synaptic weights which are used to integrate dentritic inputs x_i. If the input is strong[3] enough such that the integration exceeds the threshold $-b$, the neuron will fire. There are many other abstract models that try to more faithfully capture the complexity of a biological neuron, e. g., by describing a dynamical system rather than a function such as in the *leaky integrate-and-fire* model [29]. However, we will see that the perceptron constitutes a sufficient computational unit to construct a powerful function approximator.

For the sake of learning, we replace the perceptron's Heaviside with a nonlinear function that provides useful information to a gradient-based optimization algorithm, e. g., a sigmoidal. This function is called **activation function** $\sigma(\cdot)$ as it nonlinearly computes the output activations of the neuron. We now use the perceptron to build networks consisting of individual layers. Let $\boldsymbol{x}^{(l-1)}$ and $\boldsymbol{y}^{(l)}$ denote the input and output of the l-th layer, for $l \in \{0, \ldots, L\}$ with $\boldsymbol{x} \equiv \boldsymbol{x}^{(0)}$ and $\boldsymbol{y} \equiv \boldsymbol{y}^{(L)}$. We can define a so called **fully-connected** layer by stacking $n^{(l)}$ perceptron units that in parallel process the input $\boldsymbol{x}^{(l-1)}$, which can be written in matrix form as:[4]

$$\boldsymbol{y}^{(l)} = \sigma\left(W^{(l)}\boldsymbol{x}^{(l-1)} + \boldsymbol{b}^{(l)}\right) \tag{2.17}$$

where $W^{(l)} \in \mathbb{R}^{n^{(l)} \times n^{(l-1)}}$ and $\boldsymbol{b}^{(l)} \in \mathbb{R}^{n^{(l)}}$ are the synaptic weights and biases of layer l, and $\sigma(\cdot)$ is applied elementwise. The activations $\boldsymbol{y}^{(l)}$ of intermediate layers $0 < l < L$ are not part of the network's output and therefore called **hidden**. A network consisting only of fully-connected layers is called a multi-layer perceptron (**MLP**). There are many other neural network architectures, most notably are convolutional neural networks [CNN, 30], which have a translational invariance to local input patches and are the predominant model used in computer vision tasks. Just like MLPs, these networks are called **feed-forward**, as the computational graph is directed from inputs to outputs. Network architectures with recurrent connections for processing sequential data are called recurrent neural networks [RNN 31, 32]. RNNs are approximations to dynamical systems and can be thought

3 Note, that in practice Dale's principle is violated and synaptic weights w_i can have positive and negative signs. Thus, "strong" inputs are those that have high cosine similarity with the synaptic weight vector.

4 The nonlinearity $\sigma(\cdot)$ is often skipped in the output layer $\boldsymbol{y}^{(L)}$.

of as inducing a predictive distribution per timestep conditioned on past and current inputs [33].

In this thesis, we focus on feed-forward architectures. For mathematical convenience, we stack all the network's parameters $\{W^{(l)}, \boldsymbol{b}^{(l)}\}_{l=1}^{L}$ into a single vector $\boldsymbol{w}$. This allows us to describe the network's computation as a function $\boldsymbol{y}^{(L)} = f(\boldsymbol{x}^{(0)}; \boldsymbol{w})$, e. g., via recursive application of Eq. 2.17. Thus, by modifying the parameters $\boldsymbol{w}$ we can represent different functions. But how rich is the induced function space? Cybenko [23] and Hornik [24] famously showed that a single hidden-layer MLP ($L = 2$) can arbitrarily well approximate any continuous function if the network is sufficiently wide (i. e., $n^{(1)}$ is sufficiently large) and if $\sigma(\cdot)$ is not a polynomial. While the proof of this *universal function approximation* theorem is not constructive, in the sense, that it does not say how wide one has to make a neural network in practice for this property to hold, the empirical success of (deep) neural networks justifies their use as powerful function approximators.

2.2.1 *Efficient gradient-computation for layer-wise architecture via the chain rule*

Let's say our goal is to minimize the NLL loss $\mathcal{L}^{\text{dis}}(\boldsymbol{w})$ (Eq. 2.6) using a hypothesis space induced by fixed modelling choices and a "trainable" neural network $f(\boldsymbol{x}; \boldsymbol{w})$ (see, for instance, Eq. 2.8 or Eq. 2.10 for example loss instantiations). Assuming network parameters $\boldsymbol{w}$ are the only learnable component in this model, we can find the best hypothesis via the optimization problem $\arg\min_{\boldsymbol{w}} \mathcal{L}^{\text{dis}}(\boldsymbol{w})$. In practice, this optimization problem is almost exclusively solved via gradient descent using the chain rule:

$$\frac{\partial \mathcal{L}^{\text{dis}}(\boldsymbol{w})}{\partial \boldsymbol{y}^{(l)}} = \frac{\partial \mathcal{L}^{\text{dis}}(\boldsymbol{w})}{\partial \boldsymbol{y}^{(L)}} \frac{\partial f(\boldsymbol{x}; \boldsymbol{w})}{\partial \boldsymbol{y}^{(l)}} = \frac{\partial \mathcal{L}^{\text{dis}}(\boldsymbol{w})}{\partial \boldsymbol{y}^{(L)}} \frac{\partial \boldsymbol{y}^{(L)}}{\partial \boldsymbol{y}^{(L-1)}} \cdots \frac{\partial \boldsymbol{y}^{(l+1)}}{\partial \boldsymbol{y}^{(l)}} \tag{2.18}$$

The row vector $\delta_{\boldsymbol{y}}^{(l)} \coloneqq \frac{\partial \mathcal{L}^{\text{dis}}(\boldsymbol{w})}{\partial \boldsymbol{y}^{(l)}}$ is called the **backpropagated error**, and can be recursively computed via a single backward pass through the network $\delta_{\boldsymbol{y}}^{(l)} = \delta_{\boldsymbol{y}}^{(l+1)} \frac{\partial \boldsymbol{y}^{(l+1)}}{\partial \boldsymbol{y}^{(l)}}$ for $l < L$. Upon receiving the backpropagated error, each layer can compute weight updates locally (and in parallel), e. g.:

$$\frac{\partial \mathcal{L}^{\text{dis}}(\boldsymbol{w})}{\partial W_{ij}^{(l)}} = \delta_{\boldsymbol{y}}^{(l)} \frac{\partial \boldsymbol{y}^{(l)}}{\partial W_{ij}^{(l)}} \tag{2.19}$$

This algorithm for computing the gradient of the loss with respect to parameters $\boldsymbol{w}$ is called **backpropagation** and was introduced to the field

by Rumelhart, Hinton & Williams [34]. How those derivatives are used to compute the actual weight update $\Delta \boldsymbol{w}$ varies in practice. In the simplest case, one uses gradient descent $\Delta \boldsymbol{w} \equiv -\eta \frac{\partial \mathcal{L}^{\text{dis}}(\boldsymbol{w})}{\partial \boldsymbol{w}}$ with learning rate η, or its stochastic version where the loss function is evaluated on random subsets of $\mathcal{D}$ at each iteration.

2.3 CONTEXT-DEPENDENT GENERATION OF NETWORK PARAMETERS

In the previous section, we directly optimized the weights $\boldsymbol{w}$ of the neural network $f(\boldsymbol{x}; \boldsymbol{w})$. These parameters $\boldsymbol{w}$ induce one hypothesis that we can use for making predictions. In this section, we learn how one can represent multiple hypotheses via an auxiliary neural network.

To motivate why this might be useful, let's again consider the steering angle prediction task introduced in Sec. 2.1. Steering angles depend on the car model (e. g., due to varying turning radii). Thus, if a car manufacturer wants to develop a neural network-based steering angle predictor for its entire fleet, it has to take relevant differences between car models into account.[5] Let the car model be encoded in the context vector $\boldsymbol{e} \in \mathcal{E}$. Now, one simple solution would be to train independent networks $\boldsymbol{w}^{(\boldsymbol{e})}$ on model-specific datasets $\mathcal{D}^{(\boldsymbol{e})}$. This approach has two problems. The first one arises if the number of car models is large, in which case storing all these networks might become a concern. These large memory requirements seem intuitively unnecessary, given that all learned hypotheses essentially solve an almost identical task, i. e., steering angle prediction. This is directly linked to the second concern, that training independent networks might be unnecessarily data-hungry. Note, that each dataset $\mathcal{D}^{(\boldsymbol{e})}$ has to encompass the richness of all real-world driving scenarios. To overcome these two problems, we could simply condition a single neural network on $\boldsymbol{e}$ by providing $\boldsymbol{e}$ as additional input, e. g., we train a network $\tilde{f}(\tilde{\boldsymbol{x}}; \tilde{\boldsymbol{w}})$ with $\tilde{\boldsymbol{x}} := (\boldsymbol{x}, \boldsymbol{e})$ on a dataset $\big\{ \big((\boldsymbol{x}, \boldsymbol{e}), \boldsymbol{y} \big) \mid (\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{D}^{(\boldsymbol{e})} \text{ for } \boldsymbol{e} \in \mathcal{E} \big\}$. Now there is only one model $\tilde{\boldsymbol{w}}$ to be stored and transfer of knowledge between tasks can take place when learning to approximate some assumed ground-truth $p(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{e})$. This knowledge transfer is likely to result in a more efficient use of the available data. However, the training of independent models also had benefits compared to this approach. For instance, one could easily train a predictor for a new car model $\boldsymbol{e}_{\text{new}}$. The conditioned network $\tilde{f}\big((\boldsymbol{x}, \boldsymbol{e}); \tilde{\boldsymbol{w}} \big)$ makes such incorporation difficult and essentially requires training from

5 For the sake of illustration, we ignore that in practice steering angles can be preprocessed to be independent of a car's geometry.

scratch using the datasets of all models $\mathcal{E} \cup \{\boldsymbol{e}_{\text{new}}\}$. Now, is there a trade-off solution in which we hierarchically disentangle $\boldsymbol{e}$ and $\boldsymbol{x}$, while allowing transfer of knowledge and efficient memory use? For this, and several other purposes, **hypernetworks** have been introduced.

The term hypernetwork was only recently coined by Ha, Dai & Le [35], and generally refers to the following idea:

Definition 4 *A hypernetwork $h : \mathcal{E} \times \Psi \to \mathcal{W}$ is a neural network $h(\boldsymbol{e}; \psi)$ with inputs $\boldsymbol{e} \in \mathcal{E}$ and parameters $\psi \in \Psi$, that generates the parameters $\boldsymbol{w}$ of another neural network $f\big(\boldsymbol{x}; h(\boldsymbol{e}; \psi)\big)$.*

Ha, Dai & Le [35] portrayed multiple intriguing use-cases of hypernetworks. Most notably, they showed that with smart architectural choices, $h(\boldsymbol{e}; \psi)$ can be used to store a compressed representation of $\boldsymbol{w}$. In this thesis, hypernetworks will mostly be employed in a compressive regime such that $\dim(\boldsymbol{e}) + \dim(\psi) < \dim(\boldsymbol{w})$.

The idea of using a network to generate another network's weights is, however, much older than the term "hypernetwork" and, for instance, was already used in the work by Schmidhuber [36]. Nowadays, hypernetworks have been successfully applied to many ML research fields, including meta-learning [e. g., 37], continual learning [e. g., 33, 38–40], uncertainty modelling [e. g., 41–44] and many more.[6]

Coming back to our example, how can hypernetworks help when learning steering angle predictors for different car models $\boldsymbol{e}$? The compressed storage of all networks $\boldsymbol{w}^{(\boldsymbol{e})}$ within a single hypernetwork solves the memory issue. Furthermore, having to find shared representations within $h(\boldsymbol{e}; \psi)$ facilitates knowledge transfer between tasks. This can be easily seen by realizing that this nested setup $f\big(\boldsymbol{x}; h(\boldsymbol{e}; \psi)\big)$ is essentially a special realization of the $\boldsymbol{e}$-conditioned network $\tilde{f}\big((\boldsymbol{x}, \boldsymbol{e}); \tilde{\boldsymbol{w}}\big)$ in which case trainable parameters are $\tilde{\boldsymbol{w}} \equiv \psi$. But this special case allows overcoming some of the drawbacks $\tilde{f}\big((\boldsymbol{x}, \boldsymbol{e}); \tilde{\boldsymbol{w}}\big)$ generally has. For instance, we will show in Chapter 5 that it is now much easier to incorporate a new car model $\boldsymbol{e}_{\text{new}}$ into the hypernetwork. There are also considerable efficiency gains when deploying the trained main networks without a hypernetwork, as individual car models $\boldsymbol{e}$ do not need to recompute $\boldsymbol{w}^{(\boldsymbol{e})} = h(\boldsymbol{e}; \psi)$ for making predictions on incoming $\boldsymbol{x}$ via $f(\boldsymbol{x}; \boldsymbol{w}^{(\boldsymbol{e})})$.

This example illustrates the usefulness of hypernetworks for learning multiple tasks. But, as we will discuss in the next chapter, there are argu-

6 To further facilitate research on hypernetworks, we developed a Python package that makes it easy to explore ideas surrounding this topic: `https://github.com/chrhenning/hypnettorch`

ments that suggest that reliability and robustness can be improved also for a single task if multiple hypotheses are maintained (e. g., by inferring different estimates of the parameters $\boldsymbol{w}$ from the finite dataset $\mathcal{D}$). In this context, hypernetworks will play an important role by allowing to model complex distributions over the parameter space $\mathcal{W}$.

3

UNCERTAINTY ESTIMATION

In the previous chapter, we recapitulated the basics of the statistical framework used to describe learning in neural networks. We have seen that learning essentially amounts to estimating the parameters $\boldsymbol{w}$ of a model $p(\boldsymbol{y} \mid \boldsymbol{w}, \boldsymbol{x})$ via the dataset $\mathcal{D}$ in order to capture useful properties of some assumed ground-truth $p(\boldsymbol{y} \mid \boldsymbol{x})$ for $\boldsymbol{x} \sim p(\boldsymbol{x})$. Assuming a model[1] and using a finite sample $\mathcal{D}$ for parameter estimation is, however, troublesome as we will outline in this chapter.

As in Chapter 2, we use the example of steering angle prediction to convey concepts. Consider that such system has been trained on a dataset that does not cover snowy weather conditions. So, what will the trained model predict under such weather conditions? In principle, there are arbitrary many conditional distributions that agree with the observed dataset but lead to varying predictions in such novel situations. Since the data does not explicitly tell us how to behave there, the hypothesis space and the learning algorithm for choosing a hypothesis must be set up in a way that ensures "common sense" extrapolations. This desideratum seems impossible to fulfill by human modellers when the model is defined via a neural network. Furthermore, the neural network yields a predictive distribution $p(\boldsymbol{y} \mid \boldsymbol{w}, \boldsymbol{x})$ for any $\boldsymbol{x} \in \mathbb{R}^{n_{\text{in}}}$, independent on whether $\boldsymbol{x}$ falls in the support of $p(\boldsymbol{x})$ (or even the subspace $\mathcal{X} \subseteq \mathbb{R}^{n_{\text{in}}}$ on which $p(\boldsymbol{x})$ is defined on). This leads to the absurd situation that we can show a picture of an elephant (or just static noise) to the network, and it will make a steering angle prediction.

Ideally, we face these challenges by augmenting the learning framework from Chapter 2 to obtain models that *know what they don't know*. When discussing this desideratum, we would like to first disentangle two concepts that are too often mixed.[2] First, there might be a knowledge gap that can be solved by seeing more data (as in the case of snowy weather conditions). Second, the system might receive inputs that are not applicable to the task at hand. Note, that the ground-truth $p(\boldsymbol{y} \mid \boldsymbol{x})$ is only well defined within the support of $p(\boldsymbol{x})$. Thus, in the case of steering angle prediction, we would like to distinguish between facing novel traffic situations and having to predict

1 Which means: defining an hypothesis space (cf. Fig. 2.1).

2 We will discuss in Chapter 7 whether both subproblems can be tackled with the same formalism.

steering angles when seeing an elephant. Why is this distinction important? In the former case, it is meaningful to ask whether we can extrapolate learned knowledge to novel inputs (recall, that we distinguished learning from memorization by the ability to predict on unseen inputs in Sec. 2.1). In the latter case, it is not meaningful to generalize to unseen inputs, and indeed might be dangerous if a car makes predictions for nonsensical inputs. But as long as we do not know the support of $p(\boldsymbol{x})$, distinguishing between the two scenarios from finite observations $\mathcal{D}$ alone is practically impossible.[3] It is yet constructive to bear this distinction in mind, when assessing the solutions for uncertainty estimation discussed in this chapter.

Before we try to provide an answer for the question of how to obtain models that *know what they don't know*, we start by introducing a precise vocabulary that allows us to talk about uncertainty.

3.1 TYPES OF UNCERTAINTY

Uncertainty can be understood as the degree to which we are unable to say the following statement is true: "$\boldsymbol{y} \in \mathcal{Y}$ is the unique prediction for the input $\boldsymbol{x}$". Later in this chapter, we can be much more precise about the meaning of the term uncertainty and how it can be quantified. To do so, we first need to decide how uncertainty can be modelled. In this regard, it will prove useful if we further disentangle separate aspects of uncertainty, which is the goal of this section.

The main distinction we are going to make when talking about uncertainties will be the one commonly made between reducible and irreducible uncertainties [45, 46].

Definition 5 ***Aleatoric uncertainty*** *(latin: aleator – gambler) is irreducible uncertainty as it is the uncertainty intrinsic to the data-generating process* $p(\boldsymbol{y} \mid \boldsymbol{x})$.

Recall, that according to Def. 2, we believe in the existence of a ground-truth (cf. Sec. 2.1.3). Even if our model $p(\boldsymbol{y} \mid \boldsymbol{w}, \boldsymbol{x})$ matches this ground-truth perfectly, predictions will be uncertain as long as $p(\boldsymbol{y} \mid \boldsymbol{x})$ is not a Dirac distribution.

This uncertainty arises if $\boldsymbol{x}$ does not contain enough information for predicting $\boldsymbol{y}$. For instance, a blurry image might not allow to uniquely determine an object class. However, aleatoric uncertainty should not simply

3 While generative modelling is difficult or often infeasible, it is worth noting that it does not require labeled data. For instance, reconsidering the example from Sec. 2.3, the support of $p(\boldsymbol{x})$ can be estimated using the inputs from all datasets of different car models $\mathcal{D}^{(e)}$.

be reduced to sensory limitations (or even sensory noise). For instance, Fig. 3.1 depicts a driving scenario where the current lane splits into two. Thus, the driver has to make a decision (e. g., by following a navigation system). The input camera image $\boldsymbol{x}$, however, does not tell us how to make such decision, causing a bimodal distribution over possible steering angles $p(\boldsymbol{y} \mid \boldsymbol{x})$. Ideally, the dataset $\mathcal{D}$ is rich enough and contains many similar situations, where half of the time the driver chooses to go left and the other half chooses to go right. Thus, the ambiguity of such situations could be captured when learning a model from $\mathcal{D}$.

At this point, the discussion on model misspecification in Sec. 2.1.3 becomes crucial. In Example 4 we considered a model $p(\boldsymbol{y} \mid \boldsymbol{w}, \boldsymbol{x}) = \mathcal{N}\left(y; f(\boldsymbol{x}; \boldsymbol{w}), \sigma^2\right)$. In this case, the uncertainty is independent of the input $\boldsymbol{x}$ (we assume a *homoscedastic* noise model) and preset by the hyperparameter σ. As shown in Eq. 2.13, if $|\mathcal{D}| \to \infty$ we recover the ground-truth mean $f(\boldsymbol{x}; \boldsymbol{w}) = \mathbb{E}_{p(\boldsymbol{y}|\boldsymbol{x})}[y]$.[4] Unfortunately, capturing the ground-truth mean in Fig. 3.1 results in the fatal decision of crashing the car in between the two lanes.

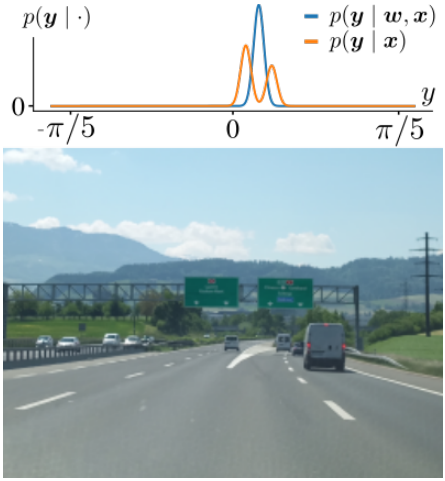


FIGURE 3.1: Figure adapted from Ref. [1]. Current input $\boldsymbol{x}$ from the front camera (bottom) shows a lane splitting scenario. This causes the ground-truth predictive distribution $p(\boldsymbol{y} \mid \boldsymbol{x})$ to be bimodal, which cannot be captured by a Gaussian model $p(\boldsymbol{y} \mid \boldsymbol{w}, \boldsymbol{x})$.

Thus, the ability to properly capture aleatoric uncertainty depends on the modelling choices [1], but also on how well we capture the ground-truth with the hypothesis chosen from the model. The NLL (Eq. 2.6) is a proper scoring rule [47] and allows us to compare hypotheses in terms of how well they are calibrated towards the ground-truth.[5] In particular, we can estimate $\mathbb{E}_{p(\boldsymbol{x})}\left[\mathrm{KL}\left(p(\boldsymbol{y} \mid \boldsymbol{x}); p(\boldsymbol{y} \mid \boldsymbol{w}, \boldsymbol{x})\right)\right]$ (up to constants) by using:

4 Assuming $f(\boldsymbol{x}; \boldsymbol{w})$ is a universal function approximator.
5 There are many other commonly used ways of measuring calibration [48, 49].

$$\mathrm{NLL}^{\mathrm{val}}(\boldsymbol{w}) \coloneqq -\sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathcal{D}^{\mathrm{val}}} \log p(\boldsymbol{y} \mid \boldsymbol{w}, \boldsymbol{x}) \tag{3.1}$$

The dataset $\mathcal{D}^{\mathrm{val}}$ denotes a validation set to control for overfitting. We say $p(\boldsymbol{y} \mid \boldsymbol{w}^{(1)}, \boldsymbol{x})$ is better calibrated than $p(\boldsymbol{y} \mid \boldsymbol{w}^{(2)}, \boldsymbol{x})$ if $\mathrm{NLL}^{\mathrm{val}}(\boldsymbol{w}^{(1)}) < \mathrm{NLL}^{\mathrm{val}}(\boldsymbol{w}^{(2)})$. Thus, by measuring calibration we can implicitly compare which hypothesis better captures aleatoric uncertainty. It is worth pointing out, that calibration as well as aleatoric uncertainties are concepts that are bound to the support of $p(\boldsymbol{x})$ where the ground-truth $p(\boldsymbol{y} \mid \boldsymbol{x})$ is well-defined.

We described aleatoric uncertainty as irreducible uncertainty, but that we can learn to capture from data. On the other hand, there is uncertainty arising due to limited knowledge:

Definition 6 ***Epistemic uncertainty*** *(greek: episteme – knowledge) is reducible uncertainty, that arises due to the fact that we only have access to finite data from which we aim to infer the ground-truth $p(\boldsymbol{y} \mid \boldsymbol{x})$.*

Epistemic uncertainty encompasses two separate aspects [46]: *model uncertainty* and *approximation uncertainty* (Fig. 3.2). Approximation uncertainty tells us how sure we are that the hypothesis h found via $\mathcal{D}$ is identical to the hypothesis $h^{(*)}$, i. e., the one closest to the ground-truth $p(\boldsymbol{y} \mid \boldsymbol{x})$ within our hypothesis space $\mathcal{H}$. For instance, h could be the maximum likelihood solution (minimum of Eq. 2.6) and $h^{(*)} = \arg\min_{h\in\mathcal{H}} \mathbb{E}_{p(\boldsymbol{x})}\big[\mathrm{KL}\big(p(\boldsymbol{y} \mid \boldsymbol{x}); h\big)\big]$ (see Eq. 2.5). As knowledge increases $|\mathcal{D}| \to \infty$, the two objectives (Eq. 2.5 and Eq. 2.6) have the same minimum, and approximation uncertainty can vanish under the assumption of perfect optimization.

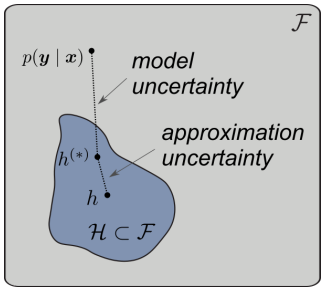


FIGURE 3.2: Figure adapted from Ref. Hüllermeier & Waegeman [46]. Epistemic uncertainty encompasses model and approximation uncertainty.

Model uncertainty, on the other hand, is the uncertainty about whether the hypothesis class $\mathcal{H}$ includes the ground truth. Hence, it is related to the concept of model misspecification (cf. Sec. 2.1.3). We have seen above, that we can use the NLL to compare hypotheses. We can also use it to compare hypotheses across different hypothesis spaces. Again, as knowledge increases

$|\mathcal{D}| \to \infty$, we can compare the NLL of the maximum likelihood solutions from different hypothesis spaces. If the considered hypothesis spaces cover the space of all hypotheses $\mathcal{F}$, then this procedure will lead to vanishing model uncertainty.

In this thesis, and in line with most of the recent literature [46], we ignore model uncertainty, such that our epistemic uncertainty estimates only capture approximation uncertainty. As a result, and also in line with most of the literature on Bayesian deep learning, we often use the term epistemic uncertainty synonymously to approximation uncertainty. We study the pitfalls of such ignorance in Ref. [1].[6] As most realistic benchmarks considered in this thesis are classification benchmarks, we hope that by the use of neural networks the hypothesis space is rather versatile as depicted in Fig. 2.1a, and that the consequences of ignoring model uncertainty are minor. However we stress that, even for the development of tools for estimating approximation uncertainty, assumptions have to be made in the rest of this chapter. And while these tools may contribute to robust and reliable uncertainty estimates in safety-critical application, they are not by themselves sufficient. Hence, they are only intended to increase (and not accomplish) trustworthiness.

3.2 CAPTURING EPISTEMIC UNCERTAINTY

> *There's only one way of doing physics but there seems to be at least two ways to do statistics, and they don't always give the same answers.*
>
> — Bradley Efron

In the previous section, we saw that if our model is appropriately chosen, aleatoric uncertainty can already be captured. Thus, in a sense, a sufficiently well calibrated model *knows what it can't know*. However, how sure can we be that we capture the ground-truth sufficiently well when only seeing limited observations $\mathcal{D}$? To capture this type of uncertainty as well, we need to augment the model with the ability to *know what it doesn't know*.

Let $\boldsymbol{w}^{(*)} \in \mathcal{W}$ denote the parameters of the unknown hypothesis from our hypothesis class that is closest to the ground-truth. Our goal is thus to estimate $\boldsymbol{w}^{(*)}$ to achieve best calibration within our hypothesis class. Since the ground-truth is unknown to us apart from observations $\mathcal{D}$, we discussed in Sec. 2.1.1 to use the MLE $\boldsymbol{w}^{\text{MLE}}$ as an estimate of $\boldsymbol{w}^{(*)}$. But how

6 For instance, if we observe vanishing approximation uncertainty in Example 4, we could only be sure that the ground-truth's mean is properly captured, but not the ground-truth itself.

good is this estimate? This is a statistical question, thus an answer should be provided by the field of statistics. Unfortunately, there are two schools of statistics that can be queried for an answer: classical (or frequentist) statistics and Bayesian statistics [50]. Yet, the deep learning literature seems to focus almost exclusively on Bayesian approaches for uncertainty estimation. This is probably because frequentist approaches to represent uncertainty over parameters (e. g., confidence intervals) are not easily translatable to predictive uncertainties, as they do not consider a joint probabilistic model as we will outline below.

Before we delve into the details of Bayesian statistics, let's build an intuition of what we seek to accomplish. In the work by Ghahramani [51] it reads "learning can be thought of as inferring plausible hypotheses to explain observed data".[7] This nicely summarizes how we want to capture epistemic (approximation) uncertainty. Essentially, instead of committing to a single hypothesis, we aim at maintaining all hypotheses that agree with the data $\mathcal{D}$ as illustrated in Fig. 3.3. Therefore, we can naturally detect for which unseen inputs $\boldsymbol{x}$ we should abstain from making predictions by looking at the disagreement between individual hypotheses. But how can we realize this conceptual idea in practice?

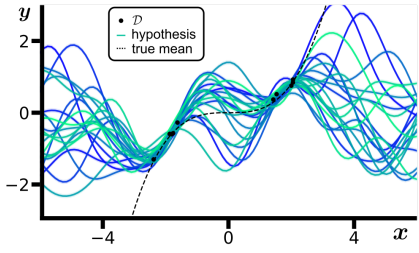


FIGURE 3.3: Hypotheses that agree with the dataset $\mathcal{D}$. The mean of the ground-truth $p(\boldsymbol{y} \mid \boldsymbol{x})$ cannot be determined with certainty from the few observed data points, especially in regions where no data has been seen at all.

Let's revisit the fundamental assumption from Def. 2 once more. We assume there is a ground-truth conditional $p(\boldsymbol{y} \mid \boldsymbol{x})$ and when having a well-specified model, we are interested in finding the parameters $\boldsymbol{w}^{(*)}$ that induce a hypothesis identical to the ground-truth. Note, that $\boldsymbol{x}, \boldsymbol{y}$ are random variables that have a frequentist probabilistic interpretation, as we can repeatedly sample from the joint $p(\boldsymbol{y} \mid \boldsymbol{x})p(\boldsymbol{x})$ to compute probabilities as frequencies (e. g., by collecting more data via a human driver).[8] The

7 We replaced the original word "models" with "hypotheses" in this quote to match our terminology.

8 This view is not shared by Levin, Tishby & Solla [52]. Indeed, considering the frequentist definition of probabilities by Cox [3], it may be questionable whether one can assume that

parameters $\boldsymbol{w}$, however, have no such interpretation, as there is only one ground-truth hypothesis in reality and no repeatable experiment can be made for drawing $\boldsymbol{w}$.

In Bayesian statistics, probabilities can also represent subjective beliefs, and thus uncertainty reflects a lack of knowledge. To apply Bayesian statistics to our model, we incorporate the parameters $\boldsymbol{w}$ as random variable into a graphical model as illustrated in Fig. 3.4 [53].[9] Now, we need to specify an additional distribution $p(\boldsymbol{w})$, the so-called **prior distribution**. As explained above, $p(\boldsymbol{w})$ does not really exist (there is only one ground-truth $\boldsymbol{w}^{(*)}$) and thus we have to choose a distribution that we use as prior. Such choice, if not uniform,[10] will induce a weighting in the hypothesis space such that some hypotheses are a priori more likely to have generated the data than others. This weighting can be considered our subjective beliefs about how the data was generated. It is an ongoing debate whether prior distributions can be chosen in a truly objective manner [4, 55].

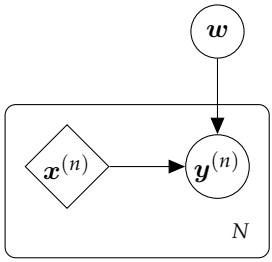


FIGURE 3.4: Probabilistic graphical model considered when applying Bayesian statistics to supervised learning.

The graphical model in Fig. 3.4 allows us to define a joint distribution over $\boldsymbol{y}$ and $\boldsymbol{w}$ induced at a point $\boldsymbol{x}$ using the likelihood $p(\boldsymbol{y} \mid \boldsymbol{w}, \boldsymbol{x})$ and prior $p(\boldsymbol{w})$. Note, that the input $\boldsymbol{x}$ is not a random variable in Fig. 3.4 such that the model only prescribes a distribution over $\boldsymbol{y}$ at an observed $\boldsymbol{x}$.[11] As the complete training set $\mathcal{D}$ is comprised in the graphical model of Fig. 3.4 and data points are conditionally independent of $\boldsymbol{w}$, we can consider the overall likelihood of parameters $\boldsymbol{w}$ as:

$$p(\mathcal{D} \mid \boldsymbol{w}) := \prod_{n=1}^{N} p(\boldsymbol{y}^{(n)} \mid \boldsymbol{w}, \boldsymbol{x}^{(n)}) \tag{3.2}$$

"indefinite repetitions" of natural images can occur. However, in our opinion, not subscribing to a frequentist interpretation of these probabilities questions the validity of Def. 2 (cf. Sec. 2.1.3).

9 Formal and (or) philosophical arguments that justify the Bayesian approach can be found in the works by Jaynes [4] and Bruinsma, Foong & Turner [54].

10 Note, that a uniform distribution in weight space does not necessarily correspond to a uniform distribution over hypotheses [4].

11 Hence, a more precise notation would be $p(\boldsymbol{y} \mid \boldsymbol{w}; \boldsymbol{x})$, which we omit for consistency.

Using Bayes' rule, we can now construct a distribution over all plausible hypotheses that explain the observed data, which is called the **posterior parameter distribution**:

$$p(\boldsymbol{w} \mid \mathcal{D}) = \frac{p(\mathcal{D} \mid \boldsymbol{w})p(\boldsymbol{w})}{p(\mathcal{D})} \tag{3.3}$$

The marginal likelihood $p(\mathcal{D}) = \mathbb{E}_{p(\boldsymbol{w})}[p(\mathcal{D} \mid \boldsymbol{w})]$, also called **model evidence**, is a quantity that can be used for model selection. Note, that our notation hides the modelling choices $\mathcal{M}$ that define the hypothesis space and the prior-induced weighting of hypotheses. Assuming no prior preference for one of the two models $\mathcal{M}_1$ and $\mathcal{M}_2$, we can compare the corresponding marginal likelihoods, $p(\mathcal{D} \mid \mathcal{M}_1)$ and $p(\mathcal{D} \mid \mathcal{M}_2)$, for selecting a hypothesis space. Since this criterion prefers a simple hypothesis space if the contained hypotheses explain the data well (hypotheses are associated with high likelihood values), it is sometimes referred to as Bayesian Occam's razor [56, 57]. We will not make use of model comparisons in this thesis and thus suppress $\mathcal{M}$ from our notation.

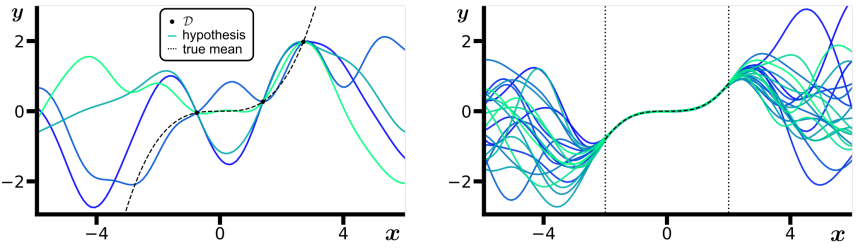
The posterior $p(\boldsymbol{w} \mid \mathcal{D})$ shrinks the set of all hypotheses induced by the prior $p(\boldsymbol{w})$ to those that explain the data well (have high likelihood values). Indeed, the hypotheses depicted in Fig. 3.3 are samples from an actual posterior distribution. Interestingly, the prior influence weakens with growing dataset size:

$$\log p(\boldsymbol{w} \mid \mathcal{D}) = \text{const.} + \log p(\boldsymbol{w}) + \sum_{n=1}^{N} \log p(\boldsymbol{y}^{(n)} \mid \boldsymbol{w}, \boldsymbol{x}^{(n)}) \tag{3.4}$$

Thus, the Bayesian formalism is biased but can be consistent and thus overcome incorrect subjective prior beliefs as $N \to \infty$ (assuming the prior has full support.).[12] The Bernstein-von-Mises theorem studies this case (under certain conditions), showing that the posterior asymptotically concentrates around the MLE [59]. Simply put, once there is no knowledge uncertainty left because we have seen everything, the posterior collapses to the MLE [60].[13] This is a quite desirable property, given that the MLE in this limit represents the hypothesis that minimizes the KL-divergence to the ground-truth $p(\boldsymbol{y} \mid \boldsymbol{x})$ (Sec. 2.1).

12 See work by Diaconis & Freedman [58] on a discussion where Bayes estimates can be inconsistent.

13 Recall, that under model misspecification the MLE is not the ground-truth (cf. Example 4 and the work by Owhadi, Scovel & Sullivan [25]).



(a) (b)

FIGURE 3.5: **(a)** Hypotheses that correspond to MLE solutions by fitting the data points perfectly. **(b)** Hypotheses that represent MAP (and MLE) solutions in the infinite data limit ($|\mathcal{D}| \to \infty$), when assuming that the support of the ground-truth $p(\boldsymbol{x})$ is restricted to the domain between the vertical dashed lines.

In the case of finite data, however, the so-called **maximum-a-posteriori (MAP)** estimate is generally different from the MLE:

$$\boldsymbol{w}^{\text{MAP}} := \arg\max_{\boldsymbol{w}} p(\boldsymbol{w} \mid \mathcal{D}) = \arg\min_{\boldsymbol{w}} \big[-\log p(\mathcal{D} \mid \boldsymbol{w}) - \log p(\boldsymbol{w}) \big] \quad (3.5)$$

Thus, the MAP is influenced by prior knowledge. Note, that the first term on the LHS in Eq. 3.5 is just the NLL (Eq. 2.6), while the second term can be seen as a regularizer punishing hypotheses that do not conform to prior beliefs. For instance, if a zero-mean Gaussian prior with unit variance is chosen, then the second term simply becomes an L2 regularization $-\log p(\boldsymbol{w}) = \text{const.} + \frac{1}{2}\|\boldsymbol{w}\|_2^2$.

Remark 1 – ***On the uniqueness of the MAP.*** *It is worth pointing out that neither the MLE $\boldsymbol{w}^{MLE}$ nor the MAP $\boldsymbol{w}^{MAP}$ are necessarily unique. If $\boldsymbol{w}$ parametrizes a neural network, then the corresponding hypothesis has trivially no unique parametrization due to permutation invariances. But also the hypothesis induced by $\boldsymbol{w}$ does not have to be unique. Fig. 3.5a shows several MLE solutions corresponding to zero mean-squared error loss (see Example 2). If these hypotheses also have the same density value under the prior $p(\boldsymbol{w})$, they can additionally be MAP solutions.*[14] *Furthermore, even in the infinite-data regime the MLE/MAP does not have to be unique if the support of $p(\boldsymbol{x})$ is bounded as shown in Fig. 3.5b. However,*

14 In certain situations, the posterior can be shown to be unimodal with unique MAP. For instance, in the conjugate setting of Gaussian process regression [61], which also applies to neural networks in the infinite-width limit [62].

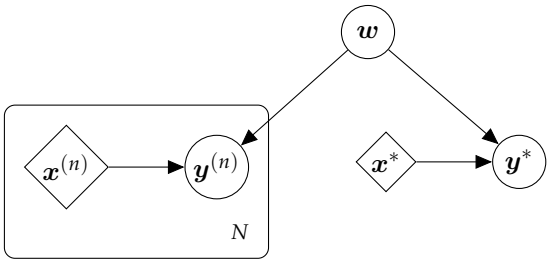


FIGURE 3.6: Extension of the probabilistic graphical model in Fig. 3.4 to incorporate an unseen test input $\boldsymbol{x}^*$.

in this case, the MLE/MAP solutions all match the ground-truth $p(\boldsymbol{y} \mid \boldsymbol{x})$ where it is well defined.

The MAP is our best estimate of the ground-truth's parameters $\boldsymbol{w}^{(*)}$ when incorporating prior knowledge. But the MAP still corresponds to a single hypothesis and thus making predictions with the MAP-induced hypothesis does not allow us to represent epistemic uncertainties (cf. Fig. 3.3).

However, incorporating the parameters $\boldsymbol{w}$ into the graphical model allows us to do probabilistic inference. In particular, for any unseen input $\boldsymbol{x}^*$, we marginalize the parameters $\boldsymbol{w}$ from the joint distribution $p(\boldsymbol{y}, \boldsymbol{w} \mid \mathcal{D}, \boldsymbol{x}^*) = p(\boldsymbol{w} \mid \mathcal{D}) p(\boldsymbol{y} \mid \boldsymbol{w}, \boldsymbol{x}^*)$ conditioned on the dataset $\mathcal{D}$ (cf. Fig. 3.6). This procedure yields the **posterior predictive distribution**:

$$p(\boldsymbol{y} \mid \mathcal{D}, \boldsymbol{x}^*) = \mathbb{E}_{p(\boldsymbol{w} \mid \mathcal{D})} \big[p(\boldsymbol{y} \mid \boldsymbol{w}, \boldsymbol{x}^*) \big] \tag{3.6}$$

This quantity is also called the Bayesian model average (BMA).[15] Note, that the posterior predictive reflects aleatoric and epistemic uncertainty. If we have seen enough data for the input $\boldsymbol{x}^*$ such that all posterior hypotheses $p(\boldsymbol{y} \mid \boldsymbol{w}, \boldsymbol{x}^*)$ for $\boldsymbol{w} \sim p(\boldsymbol{w} \mid \mathcal{D})$ are identical, then also the posterior predictive $p(\boldsymbol{y} \mid \mathcal{D}, \boldsymbol{x}^*)$ will be identical to these individual hypotheses (this is the case within the dashed lines of Fig. 3.5b). In this case, the posterior predictive only reflects aleatoric uncertainty. If, however, the individual hypotheses yield widely different predictive distributions $p(\boldsymbol{y} \mid \boldsymbol{w}, \boldsymbol{x}^*)$ for $\boldsymbol{w} \sim p(\boldsymbol{w} \mid \mathcal{D})$, then the posterior predictive will be more uncertain than individual hypotheses. This increased uncertainty corresponds to epistemic (approximation) uncertainty. We will discuss in Sec. 3.4 how epistemic uncertainty can be quantified.

15 In our terminology (cf. Chapter 2), it should rather be called *Bayesian hypotheses average*.

To summarize, we have seen in this section that Bayesian statistics can be used to model parameter uncertainty, and that this parameter uncertainty naturally translates into predictive uncertainty via Eq. 3.6.

More precisely, parameter uncertainty arises from the posterior parameter distribution $p(\boldsymbol{w} \mid \mathcal{D})$, which captures all *plausible* hypotheses given the observed data $\mathcal{D}$ and prior knowledge. Predictive disagreement between these hypotheses allows us to reflect predictive uncertainties that arise due to missing knowledge about which hypothesis to choose. This epistemic (or, more precisely, approximation) uncertainty vanishes everywhere where we observe sufficient data and where prior knowledge (in terms of choosing a hypothesis space and a prior distribution) tells us we can generalize.

3.3 INTERPRETING PREDICTIVE UNCERTAINTIES

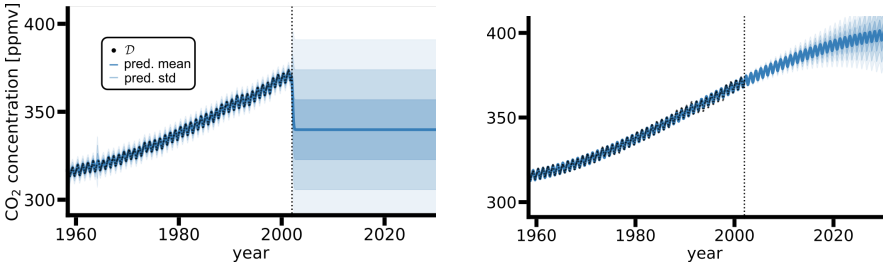
(Thanks to Francesco D'Angelo for helpful discussions on the content of this section.)

Epistemic uncertainties are difficult to interpret in practice as they heavily depend on prior knowledge, which in modern deep learning applications is chosen rather arbitrarily. To illustrate this point and emphasize that care has to be taken when interpreting uncertainties, we walk through the example depicted in Fig. 3.7 in this section.

Here, $\mathcal{D}$ is composed of atmospheric CO_2 concentrations recorded between 1958 and 2002, and we choose the same simple regression model $p(\boldsymbol{y} \mid \boldsymbol{w}, \boldsymbol{x}) = \mathcal{N}\left(y; f(\boldsymbol{x}; \boldsymbol{w}), \sigma^2\right)$ as in Example 2 for constructing a likelihood function $p(\mathcal{D} \mid \boldsymbol{w})$ (cf. Eq. 3.2). Given that this model assumes a homoscedastic noise model, we cannot learn to capture $\boldsymbol{x}$-dependent aleatoric uncertainties (cf. Sec. 3.1). Moreover, as discussed in Example 4, this modelling choice ultimately attempts to estimate the ground-truth's mean $f(\boldsymbol{x}; \boldsymbol{w}) \approx \mathbb{E}_{p(\boldsymbol{y}|\boldsymbol{x})}[y]$. Hence, uncertainties stemming from the posterior $p(\boldsymbol{w} \mid \mathcal{D})$ should reflect our missing knowledge about which $\boldsymbol{w}$ induces the ground-truth's mean via $f(\boldsymbol{x}; \boldsymbol{w})$. However, how much *knowledge is missing* depends on how much knowledge we put in a priori, as illustrated in Fig. 3.7.

In Fig. 3.7a, the prior is chosen such that mean functions $f(\boldsymbol{x}; \boldsymbol{w})$ are more likely under the prior if the corresponding function values follow the mean and variance of the CO_2 concentrations observed in $\mathcal{D}$.[16] Thus, the prior knowledge we enter into the model is not very informative about the intrinsic structure of the data. As a result, the predictive posterior

16 This is already a more careful prior construction than common choices (e. g., see Sec. 3.5) which are often made out of mathematical convenience [64, 65].



(a) uninformative prior

(b) informative prior

FIGURE 3.7: The depicted dataset $\mathcal{D}$ from Ref. [63] contains atmospheric CO_2 concentrations collected at the Mauna Loa Observatory in Hawaii until 2002 (vertical bar). The blue line depicts the mean of the posterior predictive distribution $p(\boldsymbol{y} \mid \mathcal{D}, \boldsymbol{x})$, and the shaded blue areas depict the first, second and third standard deviation of $p(\boldsymbol{y} \mid \mathcal{D}, \boldsymbol{x})$. The figure highlights that different choices of prior knowledge can completely alter the predictive distribution including predictive uncertainties, especially where no data has been seen. **(a)** A rather uninformative prior has been chosen that roughly aligns with the mean and variance of the data. **(b)** A highly-engineered prior was used (see Sec. 5.4.3 in Ref. [61]).

$p(\boldsymbol{y} \mid \mathcal{D}, \boldsymbol{x})$ can only give reasonably confident predictions in the domain where data has been seen, but becomes highly uncertain when predicting future CO_2 concentrations. In a sense, the model objectively fulfills the desideratum that it *knows what it does not know*.

On the other hand, a highly engineered prior is used to obtain the posterior predictive distribution depicted in Fig. 3.7b. In this case, the prior encodes expert knowledge about seasonal fluctuations, a long-term rising trend, etc. Now, the model can give somewhat confident predictions on how CO_2 concentrations will evolve beyond 2002. Note, that the model still *knows what it does not know*, it just has a priori more (subjective) knowledge available, which allows it to generalize past data to future predictions.

This example provides multiple interesting discussion points, that we now disentangle and look at in more detail. We should start by realizing that there is not much to learn for our model from this data. In Sec. 2.1.3 we discussed that the statistical learning framework introduced in Chapter 2 only makes intuitive sense if $\boldsymbol{x}$ contains information relevant for predicting $\boldsymbol{y}$. The atmospheric CO_2 concentration, however, does not causally depend on a point in time. Thus, the model cannot learn from the data how atmospheric CO_2 concentrations physically evolve and can only emulate

intelligent behavior based on the provided prior knowledge. Below, we try to clarify this statement.

For this example, we could easily construct a ground-truth $p_{\text{fake}}(\boldsymbol{x})p_{\text{fake}}(\boldsymbol{y} \mid \boldsymbol{x})$ that would produce a dataset similar to $\mathcal{D}$. The model does not know our abstract interpretations of $\boldsymbol{x}$ as time and $\boldsymbol{y}$ as CO_2 concentrations. So, it might as well attempt to learn any noisy function $p_{\text{fake}}(\boldsymbol{y} \mid \boldsymbol{x})$ within the support of $p_{\text{fake}}(\boldsymbol{x})$ that may emit $\mathcal{D}$ (e. g., compare Remark 1 on the uniqueness of the MLE). If the hypothesis space is rich (as for the prior in Fig. 3.7a), many hypotheses may explain the data, which causes noisy predictions in the interpolation regime (between years 1958 and 2002). By contrast, the prior chosen in Fig. 3.7b allows to locate hypotheses that give relatively consistent (and thus confident) predictions in the same period. Whether the posterior in Fig. 3.7b actually generalizes better in this period than the one in Fig. 3.7a, can only be tested with a separate validation set (e. g., cf. Eq. 3.1). If it generalizes worse, the encoded subjective prior knowledge is trivially incorrect. Thus, it would be a fallacy to believe that confident predictions under a Bayesian model are always trustworthy, especially if the prior is chosen arbitrarily. We stress this point, as such discussion or warning is often missing in the introductions of the Bayesian deep learning literature when praising the benefits of capturing epistemic uncertainty. Indeed, and as further discussed in Chapter 7, encoding prior knowledge in parametric models such as neural networks via $p(\boldsymbol{w})$ is not straightforward [66], and thus $p(\boldsymbol{w})$ is often chosen arbitrarily. Furthermore, in such complex parametric models, prior knowledge is shaped by inductive biases that are not well understood [67].

Let's reconsider the more intriguing case of forecasting CO_2 concentrations beyond 2002. The support of the training data ends here, and so, this is a case of out-of-distribution generalization [68].[17] As the data does not speak for itself in this future period, such generalization has to emerge from prior knowledge.[18] The predictive posterior in Fig. 3.7a reflects that no evidence has been seen past 2002, and thus generalization to future CO_2 concentrations is not possible. In a sense, this posterior reflects knowledge that has mostly been taken from the data since the prior is "not" informative of the data-generating process. As these priors can be seen desirable for complex problems where prior knowledge cannot easily be formalized,

17 In out-of-distribution generalization, test inputs come from a different distribution $p_{\text{test}}(\boldsymbol{x})$ than train inputs $p(\boldsymbol{x})$.

18 To fully appreciate this remark, it is important to recall that there is no reason to believe that a deeper understanding of the nature of the underlying task can emerge from the data. In this example, this can trivially not happen as CO_2 concentration are not causally related to time.

we study their existence in the context of neural networks extensively in Chapter 7. On the other hand, the prior used to obtain the posterior in Fig. 3.7b is desirable in the sense that it is more data efficient at least if prior knowledge is correct and good generalization is obtained with little data. However, it should be noted that out-of-distribution generalization cannot be tested in this case, as no validation set on future data can be collected.

In summary, this section emphasizes that predictive uncertainties that originate from a Bayesian treatment have only subjective meaning and need to be interpreted with care. This is especially relevant for high-dimensional data where it is difficult to formally encode and verify prior knowledge. While the content of this section might at first sight demotivate the Bayesian approach to learning, one should bear in mind that the learning approach presented in Chapter 2 suffers from the same shortcomings. Finding a single hypothesis that generalizes well via loss minimization also depends on prior knowledge (in terms of a well chosen hypothesis space and inductive biases intrinsic to the optimization procedure).[19] However, a single hypothesis has no means to reflect any epistemic uncertainties. Expressed in a succinct manner, we prefer the Bayesian approach because *something is better than nothing*.

3.4 QUANTIFYING EPISTEMIC UNCERTAINTY

(The content of this section originates from joint work with Francesco D'Angelo and Maria R. Cervera and can be found in SM D of Ref. [1].)

As outlined in Sec. 3.1, epistemic uncertainty consists of model and approximation uncertainty [46], and we only capture the latter by using the Bayesian approach to learning (cf. Sec. 3.2). Thus, quantifying the epistemic uncertainty considered by us amounts to quantifying the predictive uncertainty induced by the posterior parameter distribution $p(\boldsymbol{w} \mid \mathcal{D})$

Epistemic uncertainty is fundamental to the Bayesian formulation. Nevertheless, there seems to be no generally accepted agreement on how this type of uncertainty can be quantified in an input-dependent way.

A common way of quantifying uncertainty is to consider the variance or entropy of the posterior predictive distribution $p(\boldsymbol{y} \mid \mathcal{D}, \boldsymbol{x})$. However, the posterior predictive distribution does not allow for a disentanglement of

19 This remark can also be understood as an implication of the famous "no free lunch theorem" [69, 70]. Mentioning this theorem, we, however, want to express that we do believe in prior knowledge generally useful for real-world tasks as argued for by Rao, Gordon & Spears [71], and as, in our opinion, witnessed by the success of modern deep learning algorithms.

epistemic and aleatoric uncertainty and should therefore not be used as a measure of the former (cf. Sec. 3.2).

To overcome this limitation, we consider possible ways of quantification that are based on the intuition that insufficient knowledge about how to treat an input $\boldsymbol{x}^*$ is reflected by the fact that the predictive distributions of individual hypotheses $p(\boldsymbol{y} \mid \boldsymbol{w}, \boldsymbol{x}^*)$ from the posterior parameter distribution $p(\boldsymbol{w} \mid \mathcal{D})$ disagree (cf. Fig. 3.3). Thus, we aim to measure **model disagreement**.[20]

In the context of neural networks, models are usually defined as in Examples 2 and 3, where a function $f(\boldsymbol{x}; \boldsymbol{w})$ outputs the parameters ν of a distribution $p(\boldsymbol{y} \mid \boldsymbol{w}, \boldsymbol{x}) \equiv p\big(\boldsymbol{y} \mid f(\boldsymbol{x}; \boldsymbol{w})\big)$. For instance, in Example 2, ν is the mean of a Gaussian with constant variance. Whenever the mapping between these parameters $\nu = f(\boldsymbol{x}; \boldsymbol{w}) \in \mathbb{R}^{N_\nu}$ and the predictive distribution $p(\boldsymbol{y} \mid \boldsymbol{w}, \boldsymbol{x})$ is injective,[21] the aforementioned disagreement can be easily measured as the average variance across the parameter vector ν:

$$\mathcal{U}_{\mathrm{V}}^{\mathrm{epis}}(\boldsymbol{x}) = \frac{1}{N_\nu} \sum_{i=1}^{N_\nu} \mathrm{Var}_{p(\boldsymbol{w}|\mathcal{D})}[\nu_i] \tag{3.7}$$

More directly, model disagreement can be quantified via the use of a divergence, measuring the average discrepancy between the posterior predictive $p(\boldsymbol{y} \mid \mathcal{D}, \boldsymbol{x})$ and individual hypotheses $p(\boldsymbol{y} \mid \boldsymbol{w}, \boldsymbol{x})$ drawn from the posterior $p(\boldsymbol{w} \mid \mathcal{D})$:

$$\mathcal{U}_{\mathrm{D}}^{\mathrm{epis}}(\boldsymbol{x}) = \mathbb{E}_{p(\boldsymbol{w}|\mathcal{D})}\left[\mathrm{D}\left(p(\boldsymbol{y} \mid \boldsymbol{w}, \boldsymbol{x}) \mid\mid p(\boldsymbol{y} \mid \mathcal{D}, \boldsymbol{x})\right)\right] \tag{3.8}$$

Here, $\mathrm{D}(\cdot \mid\mid \cdot)$ is a statistical divergence. For instance, in Ref. [1] we consider the KL divergence and the Wasserstein distance. If $\mathcal{U}_{\mathrm{D}}^{\mathrm{epis}}(\boldsymbol{x}) = 0$, then all hypotheses captured by $p(\boldsymbol{w} \mid \mathcal{D})$ agree on how to make predictions at location $\boldsymbol{x}$.

Note, that in contrast to $\mathcal{U}_{\mathrm{V}}^{\mathrm{epis}}(\boldsymbol{x})$, epistemic uncertainty estimates $\mathcal{U}_{\mathrm{D}}^{\mathrm{epis}}(\boldsymbol{x})$ can be compared even between different model choices.

Being able to quantify epistemic uncertainty allows us to detect whether we *know* how to handle an input $\boldsymbol{x}$. Taking the considerations of Sec. 3.3 into account (correct prior knowledge encoded in $p(\boldsymbol{w})$) and assuming no model misspecification [1], we can trust a model with low epistemic uncertainty for an input $\boldsymbol{x}$ to faithfully mimic the ground-truth $p(\boldsymbol{y} \mid \boldsymbol{x})$.

20 In our terminology, the name *hypotheses disagreement* would be more suitable.

21 If there can be parametrizations $\nu_1 \neq \nu_2$ that induce the same distribution over $p(\boldsymbol{y} \mid \nu)$, model disagreement cannot reliably be computed from disagreeing parametrizations ν. See our work in Ref. [1] for details.

3.5 BAYESIAN DEEP LEARNING

As mentioned in Sec. 2.2, in most cases we implement the function $f(\boldsymbol{x}; \boldsymbol{w})$ as a neural network. Therefore, we now explain how Bayesian statistics can be applied to neural networks. In this context, when treating the network's parameters $\boldsymbol{w}$ probabilistically, one usually speaks of **Bayesian neural networks (BNNs)**. These have first been introduced by Tishby, Levin & Solla [72] and MacKay [73] and are nowadays widely studied in all fields of deep learning research.

Unfortunately, Bayesian inference in neural networks is intractable in all but the simplest cases.[22] Therefore, approximations are necessary. In this section, we will briefly review the most commonly used strategies to implement BNNs.

Ultimately, we are interested in the posterior predictive distribution $p(\boldsymbol{y} \mid \mathcal{D}, \boldsymbol{x})$ from Eq. 3.6, which is obtained by averaging the predictive distributions of individual hypotheses drawn from the posterior parameter distribution $p(\boldsymbol{w} \mid \mathcal{D})$. We can approximate this integration via Monte-Carlo using parameter sample points $\boldsymbol{w}^{(1)}, \ldots, \boldsymbol{w}^{(M)} \overset{\text{i.i.d.}}{\sim} p(\boldsymbol{w} \mid \mathcal{D})$:

$$p(\boldsymbol{y} \mid \mathcal{D}, \boldsymbol{x}) = \mathbb{E}_{p(\boldsymbol{w}|\mathcal{D})}\left[p(\boldsymbol{y} \mid \boldsymbol{w}, \boldsymbol{x})\right] \approx \frac{1}{M} \sum_{m=1}^{M} p(\boldsymbol{y} \mid \boldsymbol{w}^{(m)}, \boldsymbol{x}) \tag{3.9}$$

While such Monte-Carlo integration is de facto the standard way of approximating $p(\boldsymbol{y} \mid \mathcal{D}, \boldsymbol{x})$ [67], there are many different ways of obtaining sample points $\boldsymbol{w}^{(1)}, \ldots, \boldsymbol{w}^{(M)}$ from the posterior parameter distribution. In Sec. 3.5.1 and Sec. 3.5.2, we will discuss methods for finding a distribution that approximates the true posterior parameter distribution. This way, we can use the approximate posterior to obtain an arbitrary number of sample points for the Monte-Carlo integration above. Alternatively, a direct sampling strategy is discussed in Sec. 3.5.3, which asymptotically yields sample points from the true posterior parameter distribution.

3.5.1 *Variational inference*

In this section, we explain how variational inference (VI) can be applied to obtain an approximate posterior. For a detailed introduction to VI see work

22 For instance, a linear neural network, reparametrized to have no hidden layers, with Gaussian prior can be seen as a Gaussian linear system [74], and the posterior predictive distribution has a closed-form solution.

by Blei, Kucukelbir & McAuliffe [75]. In VI, probabilistic inference is turned into optimization. More precisely, rather than computing the posterior parameter distribution $p(\boldsymbol{w} \mid \mathcal{D})$ via Eq. 3.3, we use optimization to find the parameters θ of a distribution $q(\boldsymbol{w};\theta)$ which is closest to $p(\boldsymbol{w} \mid \mathcal{D})$.[23] In our case, "closest" will be defined in the sense of the reverse KL-divergence $\mathrm{KL}\big(q(\boldsymbol{w};\theta); p(\boldsymbol{w} \mid \mathcal{D})\big)$, which allows us to derive a tractable optimization criterion as outlined below. This type of VI is the one most often found in the Bayesian deep learning literature (e. g., see works by Graves [76] and Blundell *et al.* [77] for early applications), and will be our method of choice in Chapter 6. However, other choices of divergences or distances are possible [e. g., 78].

A tractable optimization criterion can be derived as follows:

$$\begin{aligned}
\log p(\mathcal{D}) &= \mathbb{E}_{q(\boldsymbol{w};\theta)}\left[\log p(\mathcal{D})\right] && (3.10)\\
&= \mathbb{E}_{q(\boldsymbol{w};\theta)}\left[\log\left(p(\mathcal{D})\frac{q(\boldsymbol{w};\theta)}{q(\boldsymbol{w};\theta)}\frac{p(\boldsymbol{w} \mid \mathcal{D})}{p(\boldsymbol{w} \mid \mathcal{D})}\right)\right] && (3.11)\\
&= \mathbb{E}_{q(\boldsymbol{w};\theta)}\left[\log\left(\frac{p(\boldsymbol{w},\mathcal{D})}{q(\boldsymbol{w};\theta)}\frac{q(\boldsymbol{w};\theta)}{p(\boldsymbol{w} \mid \mathcal{D})}\right)\right] && (3.12)\\
&= \underbrace{\mathbb{E}_{q(\boldsymbol{w};\theta)}\left[\log\frac{p(\boldsymbol{w},\mathcal{D})}{q(\boldsymbol{w};\theta)}\right]}_{:=\mathrm{ELBO}} + \mathrm{KL}\left(q(\boldsymbol{w};\theta); p(\boldsymbol{w} \mid \mathcal{D})\right) && (3.13)
\end{aligned}$$

The first term on the LHS is called **evidence lower-bound (ELBO)**.[24] By realizing that the model evidence $\log p(\mathcal{D})$ is constant in our case (cf. Sec. 3.2) and that the KL is non-negative, we see that by maximizing the ELBO we minimize the KL-divergence from $p(\boldsymbol{w} \mid \mathcal{D})$ to $q(\boldsymbol{w};\theta)$:

$$\underbrace{\mathrm{KL}\left(q(\boldsymbol{w};\theta); p(\boldsymbol{w} \mid \mathcal{D})\right)}_{\geq 0} = \text{const.} - \mathrm{ELBO} \tag{3.14}$$

The fact that the ELBO can yield a tractable optimization criterion can be seen as follows:

$$\begin{aligned}
\mathrm{ELBO} &= \mathbb{E}_{q(\boldsymbol{w};\theta)}\left[\log\frac{p(\boldsymbol{w})p(\mathcal{D} \mid \boldsymbol{w})}{q(\boldsymbol{w};\theta)}\right] && (3.15)\\
&= \mathbb{E}_{q(\boldsymbol{w};\theta)}\left[\log p(\mathcal{D} \mid \boldsymbol{w})\right] - \mathrm{KL}\left(q(\boldsymbol{w};\theta); p(\boldsymbol{w})\right) && (3.16)
\end{aligned}$$

23 In the coming chapters, we interchangeably use the notations $q_\theta(\boldsymbol{w}) := q(\boldsymbol{w};\theta)$ to denote a distribution over $\boldsymbol{w}$ parametrized by θ.

24 In statistical physics, the negative ELBO is also called *variational free energy.*

Thus, maximizing the ELBO corresponds to maximizing the expected log-likelihood of the data $\mathcal{D}$ while staying close to the prior. Or in other words, we seek to find θ that simultaneously minimizes the expected NLL and the KL-divergence from $p(\boldsymbol{w})$ to $q(\boldsymbol{w};\theta)$:

$$\theta^{(*)} = \arg\min_{\theta} \mathbb{E}_{q(\boldsymbol{w};\theta)} \left[-\sum_{n=1}^{N} \log p(\boldsymbol{y}^{(n)} \mid \boldsymbol{w}, \boldsymbol{x}^{(n)}) \right] + \mathrm{KL}\left(q(\boldsymbol{w};\theta); p(\boldsymbol{w})\right) \tag{3.17}$$

Importantly, the above optimization criterion does not contain the intractable posterior $p(\boldsymbol{w} \mid \mathcal{D})$.[25]

The expected value in the first term of the above optimization problem is often approximated via Monte-Carlo by using the reparametrization trick [79]. Note, if the optimization problem would only consist of this first term, we would seek to find a $q(\boldsymbol{w};\theta)$ that allocates all its mass around MLE solutions.

The second term, on the other hand, pushes the approximate posterior $q(\boldsymbol{w};\theta)$ to be close to the prior $p(\boldsymbol{w})$. This term can be problematic in practice when seeking a tractable optimization criterion. To understand why, let's briefly discuss the choice of $q(\boldsymbol{w};\theta)$.

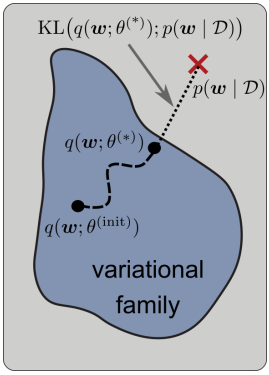


FIGURE 3.8: This figure illustrates variational inference. It depicts an optimization trajectory from $\theta^{(init)}$ to $\theta^{(*)}$, where $\theta^{(*)}$ induces the family member closest to the posterior.

The distribution $q(\boldsymbol{w};\theta)$ is taken from a family of distributions $\mathcal{Q}$ which is implicitly defined by the parameter space $\theta \in \Theta$. This family is called **variational family**. Variational inference can thus be understood as picking the distribution $q(\boldsymbol{w};\theta^{(*)})$ within this family that is closest to $p(\boldsymbol{w} \mid \mathcal{D})$. This process is illustrated in Fig. 3.8. Intriguingly, there is a trade-off between the expressiveness of the variational family and the ease of optimization. Ideally, we would like to

25 Note, the optimization problem in Eq. 3.17 is conceptually different from the one in, for instance, Eq. 2.6, which justifies why parameter uncertainty for the estimation of $\theta^{(*)}$ is not considered. The optimum of Eq. 3.17 represents the $q(\boldsymbol{w};\theta)$ closest to $p(\boldsymbol{w} \mid \mathcal{D})$ (in the KL-sense). The optimum of Eq. 2.6 represents the MLE, which is only a surrogate for the ground-truth parameters that we cannot estimate with certainty from the finite sample $\mathcal{D}$.

choose $\mathcal{Q}$ to contain $p(\boldsymbol{w} \mid \mathcal{D})$. If it is unknown how to choose such family, it may be intuitively preferable to choose $\mathcal{Q}$ as large as possible. However, in practice, $\mathcal{Q}$ is often chosen to obtain a tractable optimization criterion (Eq. 3.17). Moreover, also the prior $p(\boldsymbol{w})$ is often chosen out of convenience with respect to Eq. 3.17 rather than to reflect meaningful prior knowledge (cf. Sec. 3.3).

For instance, if both $q(\boldsymbol{w};\theta)$ and $p(\boldsymbol{w})$ are Gaussian, the second term in Eq. 3.17 has a closed-form expression. Note, that in order to maximize the ELBO via gradient-based optimization (also called *black-box variational inference* [80]), we need to obtain an estimate of the gradient $\nabla_\theta \mathrm{KL}\left(q(\boldsymbol{w};\theta); p(\boldsymbol{w})\right)$. In the case of two Gaussian distributions, this gradient can be computed analytically [e. g., 77]. More expressive variational families might require approximations to this gradient [e. g., 81, 82]. In Chapter 6, we will explore a variety of variational families and empirically observe, that simpler variational families often yield better performance when the underlying task complexity increases, most likely due to ease of optimization.[26]

3.5.2 *Laplace approximation*

(Thanks to Maria R. Cervera for helpful discussions on the content of this section.)

A Laplace approximation $q(\boldsymbol{w};\theta)$ is a Gaussian approximation of the posterior obtained via a second-order Taylor expansion around a MAP solution [73].

In particular, our approximate posterior is of the form:

$$q(\boldsymbol{w};\theta) \equiv \mathcal{N}(\boldsymbol{w}; \mu^{\mathrm{LA}}, \Sigma^{\mathrm{LA}}) \tag{3.18}$$

where $\theta = \{\mu^{\mathrm{LA}}, \Sigma^{\mathrm{LA}}\}$.

The first step in computing a Laplace approximation consists of finding a MAP estimate $\boldsymbol{w}^{\mathrm{MAP}}$ according to Eq. 3.5. The MAP determines the mean $\mu^{\mathrm{LA}} \equiv \boldsymbol{w}^{\mathrm{MAP}}$ of our approximation and the covariance matrix is computed via a second-order Taylor expansion around the MAP:

$$\log p(\boldsymbol{w} \mid \mathcal{D}) \approx \text{const.} + \frac{1}{2}(\boldsymbol{w} - \boldsymbol{w}^{\mathrm{MAP}})^T H_{\log p(\boldsymbol{w}|\mathcal{D})}\Big|_{\boldsymbol{w}=\boldsymbol{w}^{\mathrm{MAP}}}(\boldsymbol{w} - \boldsymbol{w}^{\mathrm{MAP}}) \tag{3.19}$$

26 Such statements should be taken with care, as one cannot compare to results obtained under exact Bayesian inference. In other words, "better" is a qualitative statement regarding test set performance, and is not tantamount to improved approximate inference.

Here, $H_{\log p(\boldsymbol{w}|\mathcal{D})}$ denotes the Hessian $\nabla_{\boldsymbol{w}} \nabla_{\boldsymbol{w}}^T \log p(\boldsymbol{w} \mid \mathcal{D})$. Note, that the first-order term is zero around the MAP (extremum of function $\log p(\boldsymbol{w} \mid \mathcal{D})$).

Eq. 3.19 resembles the log-probability density function of a multivariate Gaussian distribution if $(\Sigma^{\text{LA}})^{-1} \equiv -H_{\log p(\boldsymbol{w}|\mathcal{D})}\Big|_{\boldsymbol{w}=\boldsymbol{w}^{\text{MAP}}}$. However, how to compute the Hessian of the log-posterior?

First, we should realize that the Hessian can be written as:

$$H_{\log p(\boldsymbol{w}|\mathcal{D})} = H_{\log p(\mathcal{D}|\boldsymbol{w})} + H_{\log p(\boldsymbol{w})} \tag{3.20}$$

We assume that the Hessian of the prior can be computed in closed-form. For the sake of simplicity, we take $p(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}; 0, \sigma^2_{\text{prior}} I)$ such that $\log p(\boldsymbol{w}) = \text{const.} - \frac{1}{2\sigma^2_{\text{prior}}} \|\boldsymbol{w}\|_2^2$. Noting that $\nabla_{\boldsymbol{w}} \nabla_{\boldsymbol{w}}^T \|\boldsymbol{w}\|_2^2 = 2I$ we get $H_{\log p(\boldsymbol{w})} = -\frac{1}{\sigma^2_{\text{prior}}} I$.[27]

Furthermore, we can write the Hessian of the log-likelihood as:

$$H_{\log p(\mathcal{D}|\boldsymbol{w})} = \sum_{n=1}^{N} H_{\log p(\boldsymbol{y}^{(n)}|\boldsymbol{w},\boldsymbol{x}^{(n)})} \tag{3.21}$$

If $p(\boldsymbol{y} \mid \boldsymbol{w}, \boldsymbol{x})$ is defined via a neural network, computing these Hessian terms might be computationally infeasible. Therefore, further approximations are commonly employed. To understand these, we should first observe that the term $(1/N) H_{\log p(\mathcal{D}|\boldsymbol{w})}$ can be seen as a Monte-Carlo estimate of $\mathbb{E}_{p(\boldsymbol{x})p(\boldsymbol{y}|\boldsymbol{x})}[H_{\log p(\boldsymbol{y}|\boldsymbol{w},\boldsymbol{x})}]$. This observation will become useful below.

First, let's recall that via simple calculations one can show that:

$$\mathbb{E}_{p(\boldsymbol{y}|\boldsymbol{w},\boldsymbol{x})}[H_{\log p(\boldsymbol{y}|\boldsymbol{w},\boldsymbol{x})}] = \tag{3.22}$$

$$-\underbrace{\mathbb{E}_{p(\boldsymbol{y}|\boldsymbol{w},\boldsymbol{x})}[\nabla_{\boldsymbol{w}} \log p(\boldsymbol{y} \mid \boldsymbol{w}, \boldsymbol{x}) \nabla_{\boldsymbol{w}} \log p(\boldsymbol{y} \mid \boldsymbol{w}, \boldsymbol{x})^T]}_{:=F} \tag{3.23}$$

The matrix F is the Fisher information matrix. If for every training input $\boldsymbol{x}^{(n)}$ we sample a prediction via the model $\tilde{\boldsymbol{y}}^{(n)} \sim p(\boldsymbol{y} \mid \boldsymbol{w}, \boldsymbol{x}^{(n)})$, we can also obtain a Monte-Carlo estimate of the expected Fisher information matrix:

27 Note, that we again choose the prior out of mathematical convenience and not to reflect meaningful prior knowledge according to Sec. 3.3.

$$\mathbb{E}_{p(\boldsymbol{x})p(\boldsymbol{y}|\boldsymbol{w},\boldsymbol{x})}[H_{\log p(\boldsymbol{y}|\boldsymbol{w},\boldsymbol{x})}] = -\mathbb{E}_{p(\boldsymbol{x})}[F] \tag{3.24}$$

$$\approx -\frac{1}{N}\sum_{n=1}^{N} \nabla_{\boldsymbol{w}} \log p(\tilde{\boldsymbol{y}}^{(n)} \mid \boldsymbol{w}, \boldsymbol{x}^{(n)}) \nabla_{\boldsymbol{w}} \log p(\tilde{\boldsymbol{y}}^{(n)} \mid \boldsymbol{w}, \boldsymbol{x}^{(n)})^T \tag{3.25}$$

Now, if we assume that at the MAP the predictive distribution $p(\boldsymbol{y} \mid \boldsymbol{w}^{\text{MAP}}, \boldsymbol{x})$ closely resembles the ground-truth $p(\boldsymbol{y} \mid \boldsymbol{x})$, we can approximate the expected Fisher using actual data targets $\boldsymbol{y}^{(n)}$ from $\mathcal{D}$ rather than those obtained from the model $\tilde{\boldsymbol{y}}^{(n)}$.[28] This approximation of the expected Fisher is called the empirical Fisher:

$$F^{\text{emp}} := \frac{1}{N}\sum_{n=1}^{N} \nabla_{\boldsymbol{w}} \log p(\boldsymbol{y}^{(n)} \mid \boldsymbol{w}, \boldsymbol{x}^{(n)}) \nabla_{\boldsymbol{w}} \log p(\boldsymbol{y}^{(n)} \mid \boldsymbol{w}, \boldsymbol{x}^{(n)})^T \tag{3.26}$$

$$\approx \mathbb{E}_{p(\boldsymbol{x})}[F] \tag{3.27}$$

Note, that this approximation is not strictly necessary, as we usually can sample targets $\tilde{\boldsymbol{y}}^{(n)}$ from the model. However, we anyway need the assumption that $p(\boldsymbol{y} \mid \boldsymbol{w}^{\text{MAP}}, \boldsymbol{x})$ is close to $p(\boldsymbol{y} \mid \boldsymbol{x})$ within the support of $p(\boldsymbol{x})$ again to build the following chain of connections:

$$-F^{\text{emp}} \approx -\mathbb{E}_{p(\boldsymbol{x})}[F] = \mathbb{E}_{p(\boldsymbol{x})p(\boldsymbol{y}|\boldsymbol{w}^{\text{MAP}},\boldsymbol{x})}[H_{\log p(\boldsymbol{y}|\boldsymbol{w}^{\text{MAP}},\boldsymbol{x})}] \tag{3.28}$$

$$\overset{(!)}{\approx} \mathbb{E}_{p(\boldsymbol{x})p(\boldsymbol{y}|\boldsymbol{x})}[H_{\log p(\boldsymbol{y}|\boldsymbol{w}^{\text{MAP}},\boldsymbol{x})}] \approx \frac{1}{N} H_{\log p(\mathcal{D}|\boldsymbol{w})} \tag{3.29}$$

The crucial approximation, marked with (!), allows us to build a connection between the Hessian of the log-likelihood $H_{\log p(\mathcal{D}|\boldsymbol{w})}$ and the Fisher. Thus, we can approximate the Hessian only via first-order gradient information. The above series of approximations allows us to write the negative precision matrix as follows:

28 Note, that this is somehow a weird assumption given that we motivated Bayesian statistics by stating that a single parametrization obtained from the limited data is probably not capturing the ground-truth $p(\boldsymbol{y} \mid \boldsymbol{x})$ well. We, unfortunately, are not aware of any good justification. On the contrary, approximating the Fisher this way has been shown to not capture second-order information well [83].

$$H_{\log p(\boldsymbol{w}|\mathcal{D})}\Big|_{\boldsymbol{w}=\boldsymbol{w}^{\text{MAP}}} \approx -N\mathbb{E}_{p(\boldsymbol{x})}[F] - \frac{1}{\sigma^2_{\text{prior}}} I \tag{3.30}$$

$$\approx -NF^{\text{emp}} - \frac{1}{\sigma^2_{\text{prior}}} I \tag{3.31}$$

where the Fisher F and the empirical Fisher F^{emp} are computed using the gradient of the model evaluated at the MAP: $\nabla_{\boldsymbol{w}} \log p(\boldsymbol{y} \mid \boldsymbol{w}, \boldsymbol{x})\big|_{\boldsymbol{w}=\boldsymbol{w}^{\text{MAP}}}$.

Hence, we can write:

$$q(\boldsymbol{w};\theta) = \mathcal{N}\left(\boldsymbol{w}; \boldsymbol{w}^{\text{MAP}}, \left[NF^{\text{emp}} + \frac{1}{\sigma^2_{\text{prior}}} I\right]^{-1}\right) \tag{3.32}$$

In practice, a further approximation is often employed by only considering the diagonal of the (empirical) Fisher matrix [e. g., 84], or other simplifying structures that avoid storing a dense matrix [e. g., 85].

3.5.3 *Markov chain Monte Carlo*

In the previous two subsections we discussed ways for obtaining an approximate posterior $q(\boldsymbol{w};\theta) \approx p(\boldsymbol{w} \mid \mathcal{D})$. However, as mentioned earlier, we are mainly interested in $q(\boldsymbol{w};\theta)$ for obtaining sample points that allow Monte-Carlo integration as in Eq. 3.9.

In this section, we will briefly introduce a framework for obtaining $\boldsymbol{w}^{(1)}, \ldots, \boldsymbol{w}^{(M)}$ sample points which asymptotically can be considered as being sampled from $p(\boldsymbol{w} \mid \mathcal{D})$. This framework is called *Markov chain Monte Carlo* (MCMC) [for an introduction, see 86]. In this case, a Markov chain is simulated whose stationary distribution is set to be the posterior $p(\boldsymbol{w} \mid \mathcal{D})$.

Here, we consider the parameters $\boldsymbol{w}$ to be the states of the chain, and use the conditional $\pi(\boldsymbol{w}^{(t)} \mid \boldsymbol{w}^{(t-1)})$ to define the likelihood of moving from state $\boldsymbol{w}^{(t-1)}$ to state $\boldsymbol{w}^{(t)}$. A sufficient condition for showing that a stationary distribution $\pi(\boldsymbol{w}^{(t)})$ exists, is by showing that the chain fulfills the *detailed balance* criterion:

$$\pi(\boldsymbol{w}^{(t-1)})\pi(\boldsymbol{w}^{(t)} \mid \boldsymbol{w}^{(t-1)}) = \pi(\boldsymbol{w}^{(t)})\pi(\boldsymbol{w}^{(t-1)} \mid \boldsymbol{w}^{(t)}) \tag{3.33}$$

Our goal will be to construct the chain such that $\pi(\boldsymbol{w}^{(t)}) \equiv p(\boldsymbol{w}^{(t)} \mid \mathcal{D})$. Furthermore, we split the likelihood of moving from one state to another into two parts:

$$\pi(\boldsymbol{w}^{(t)} \mid \boldsymbol{w}^{(t-1)}) = \pi^{Q}(\boldsymbol{w}^{(t)} \mid \boldsymbol{w}^{(t-1)})\pi^{T}(\boldsymbol{w}^{(t)} \mid \boldsymbol{w}^{(t-1)}) \tag{3.34}$$

The distribution $\pi^{Q}(\boldsymbol{w}^{(t)} \mid \boldsymbol{w}^{(t-1)})$ is called the *proposal distribution*, and used to propose a new state $\boldsymbol{w}^{prop} \sim \pi^{Q}(\boldsymbol{w}^{prop} \mid \boldsymbol{w}^{(t-1)})$. This new state is accepted ($\boldsymbol{w}^{(t)} \equiv \boldsymbol{w}^{prop}$) or rejected ($\boldsymbol{w}^{(t)} \equiv \boldsymbol{w}^{(t-1)}$) with the *transition probability* $\pi^{T}(\boldsymbol{w}^{prop} \mid \boldsymbol{w}^{(t-1)})$.

Using detailed balance, we see that:

$$\frac{\pi^{T}(\boldsymbol{w}^{(t)} \mid \boldsymbol{w}^{(t-1)})}{\pi^{T}(\boldsymbol{w}^{(t-1)} \mid \boldsymbol{w}^{(t)})} = \frac{p(\boldsymbol{w}^{(t)} \mid \mathcal{D})}{p(\boldsymbol{w}^{(t-1)} \mid \mathcal{D})} \frac{\pi^{Q}(\boldsymbol{w}^{(t-1)} \mid \boldsymbol{w}^{(t)})}{\pi^{Q}(\boldsymbol{w}^{(t)} \mid \boldsymbol{w}^{(t-1)})} \tag{3.35}$$

This is fulfilled by the *Metropolis criterion* [87, 88]:

$$\pi^{T}(\boldsymbol{w}^{(t)} \mid \boldsymbol{w}^{(t-1)}) = \min\left[1, \frac{p(\boldsymbol{w}^{(t)} \mid \mathcal{D})}{p(\boldsymbol{w}^{(t-1)} \mid \mathcal{D})} \frac{\pi^{Q}(\boldsymbol{w}^{(t-1)} \mid \boldsymbol{w}^{(t)})}{\pi^{Q}(\boldsymbol{w}^{(t)} \mid \boldsymbol{w}^{(t-1)})}\right] \tag{3.36}$$

Note, if we choose the proposal to be symmetric, the Metropolis criterion can be computed by evaluating the posterior ratio. Luckily, the intractable model evidence $p(\mathcal{D})$ cancels out in this ratio leaving:

$$\frac{p(\boldsymbol{w}^{(t)} \mid \mathcal{D})}{p(\boldsymbol{w}^{(t-1)} \mid \mathcal{D})} = \frac{p(\mathcal{D} \mid \boldsymbol{w}^{(t)})p(\boldsymbol{w}^{(t)})}{p(\mathcal{D} \mid \boldsymbol{w}^{(t-1)})p(\boldsymbol{w}^{(t-1)})} \tag{3.37}$$

Thus, for computing the Metropolis criterion we need to be able to evaluate the likelihood and prior.

Algorithm 1: Metropolis-Hastings algorithm

Propose new state $\boldsymbol{w}^{prop} \sim \pi^{Q}(\boldsymbol{w}^{prop} \mid \boldsymbol{w}^{(t-1)})$;
Compute transition probability: $\pi^{T}(\boldsymbol{w}^{prop} \mid \boldsymbol{w}^{(t-1)})$ via Eq. 3.36;
Sample $u \sim U(0, 1)$ uniformly;
if $u \leq \pi^{T}(\boldsymbol{w}^{prop} \mid \boldsymbol{w}^{(t-1)})$ **then**
  $\boldsymbol{w}^{(t)} \leftarrow \boldsymbol{w}^{prop}$; `// accept`
else
  $\boldsymbol{w}^{(t)} \leftarrow \boldsymbol{w}^{(t-1)}$; `// reject`
end

The complete algorithm for simulating the chain is sketched in Alg. 1. A simple example of a proposal would be a Gaussian $\pi^{Q}(\boldsymbol{w}^{(t)} \mid \boldsymbol{w}^{(t-1)}) = \mathcal{N}(\boldsymbol{w}^{(t)}; \boldsymbol{w}^{(t-1)}, \Sigma)$ with fixed covariance matrix Σ. Such instantiation is

called *random walk MCMC*, because new states are chosen randomly in the vicinity of $\boldsymbol{w}^{(t-1)}$ which typically causes either a high rejection rate by the Metropolis criterion (Eq. 3.36) or slow exploration of the state space if new proposals are too close to previous ones.

Technically, the proposal has to be chosen such that the chain is *ergodic*, by showing it is *irreducible* and *aperiodic* [for details see 89]. If that is the case, the stationary distribution obtained after convergence is unique, and due to the construction above, it corresponds to the posterior.

As the stationary distribution is only obtained once the chain converged, the sample points obtained at the beginning are often discarded (*burn-in*).

To avoid the above mentioned random walk behavior, the proposal distribution can, for instance, incorporate gradient information $\nabla_{\boldsymbol{w}} p(\mathcal{D} \mid \boldsymbol{w}) p(\boldsymbol{w})$ for generating a proposal state. The most prominent algorithm along these lines is arguably **Hamiltonian Monte Carlo** [HMC 90, 91]. HMC is often considered the best available method for performing approximate inference in terms of obtaining high-fidelity sample points from the true posterior. For this reason, we will use this method in Chapter 7 when investigating the uncertainty properties of the true posterior. However, plain HMC is of little practical relevance due to its computational cost (even though modern compute allows its application to computer vision problems [92]). This cost mainly arises when computing $p(\mathcal{D} \mid \boldsymbol{w})$ for the Metropolis criterion which requires processing the full dataset at every step of the chain. To overcome this limitation, stochastic MCMC versions have been developed in recent years [e. g., 93, 94].

In summary, MCMC methods provide a compelling way of obtaining sample points from the real posterior. To apply these algorithms, convergence of the chain needs to be ensured and the chain needs to be run long enough such that autocorrelations between sample points are diluted enough to justify the use snapshot sample points from the chain for a Monte-Carlo integration as in Eq. 3.9. By contrast, the aforementioned methods for obtaining an approximate posterior $q(\boldsymbol{w}; \theta)$ allow to directly obtain an i.i.d. sample. However, it is in practice difficult to tell how much such sample resembles one that was obtained from $p(\boldsymbol{w} \mid \mathcal{D})$.

3.6 CLOSING REMARKS ON UNCERTAINTY ESTIMATION

In this chapter, we discussed how a model can *know what it does not know*. In Sec. 3.1 we divided uncertainty into aleatoric uncertainty, which is intrinsic to the data-generating process, and epistemic uncertainty, that can

be reduced by seeing more data and choosing the model properly (note, that model selection can be guided by the data; Sec. 3.2). We then decided to only capture part of the epistemic uncertainty, namely approximation uncertainty, by using Bayesian statistics (cf. Sec. 3.1 and Sec. 3.2). This led to the somewhat absurd situation that *knowing what we don't know* is measured relative to subjective prior knowledge (cf. Sec. 3.3). This prior knowledge encompasses all our modelling choices (including the weight prior $p(\boldsymbol{w})$), and there is no uncertainty associated with it (unless model uncertainty is considered too [46]). Furthermore, we saw in Sec. 3.5 that approximate inference is necessary, and that such methods often require us to choose prior knowledge out of mathematical (or computational) convenience.

All these complex considerations may lead to the premature conclusion that Bayesian statistics is of little use given the complexity of the problems tackled with deep learning approaches. However, it should be stressed that the field of Bayesian deep learning is a relatively young field of research, and while the tools that are being developed may often be of little practical use yet [e. g., 49], they are a necessary step to facilitate further innovation and progress in the field. As mentioned in Sec. 3.3, it is important to bear in mind that the learning approach presented in Chapter 2 suffers just as well from arbitrarily[29] chosen prior knowledge, with the difference that no uncertainty about remaining degrees of freedom is quantified. Research on how to understand (often implicitly) encoded prior knowledge in neural networks and how to encode useful prior knowledge is currently in progress [e. g., 66, 95]. At the same time, methods for improved approximate inference are constantly being developed. Furthermore, there are other theoretical benefits that come out of a Bayesian treatment as we will discuss in the next chapter. Overall, we are excited about the future prospects of the field, while trying to be careful with our formulations, for instance, by abstaining from recommending current Bayesian deep learning approaches for safety-critical applications.

29 "Arbitrary" is an exaggeration given that deep learning research is application-oriented and model selection by researchers acts like a genetic algorithm, where only the fittest survive.

4

CONTINUAL LEARNING

So far, we studied learning problems where we were given access to a prerecorded dataset $\mathcal{D}$ from which we should infer how to make predictions $\boldsymbol{y}$ on unseen inputs $\boldsymbol{x}$. In this classic view of Machine Learning, there is a training phase and a subsequent testing phase. The model is learnt via $\mathcal{D}$ during the training phase and is subsequently used for making predictions. However, once the training phase has been completed, no new evidence can be incorporated. This is in stark contrast to our intuitive understanding of learning as a lifelong process. In fact, the ability to continually learn becomes desirable whenever an existing model could be improved by new data. For instance, if new driving data from rare weather conditions becomes available or a novel street sign should be incorporated into an object classifier. Therefore, we devote this chapter to define the problem of *Continual Learning* (**CL**, see the work by Parisi *et al.* [96] for a review) and discuss different strategies to approach it.

Before diving into the problem, it is valid to ask, whether we could just retrain the model on all past and new data whenever we encounter new data (i. e., repeat the full training phase). In many cases, this is a viable approach, but one that is arguably wasteful regarding an efficient use of resources. Modern AI applications, such as large language models [e. g., 97], can require considerable compute resources for training which are associated with high energy demands and immense financial budgets [98]. Thus, avoiding to repeat training phases from scratch may lead to ecological and economic benefits, and may make applications available to a wider audience. In other cases, past data may not even be available anymore for training due to storage costs or storage location, or due to privacy concerns. Thus, in such scenarios CL algorithms are necessary for improving models. Unfortunately, state-of-the-art approaches to CL are not competitive to simple retraining on many real-world applications [40, 99]. This chapter should shed some light on why CL is such a difficult problem. Before doing so, let's try to build an intuition for why CL is not naturally possible in the learning framework studied thus far.

In Def. 2 we required the dataset $\mathcal{D}$ to be an i.i.d. sample of the ground-truth $p(\boldsymbol{x})p(\boldsymbol{y} \mid \boldsymbol{x})$. This allowed us to form an unbiased consistent estimator of a gradient that minimizes a divergence from members of our

hypothesis class to the ground-truth (e. g., via Eq. 2.6).[1] Moreover, popular algorithms such as SGD are theoretically justified through the use of independent (small) i.i.d. samples from the ground-truth $p(\boldsymbol{x})p(\boldsymbol{y} \mid \boldsymbol{x})$ [100]. These i.i.d. assumptions are heavily violated in CL scenarios as we will see later, which leads to an effect called **catastrophic interference** or **forgetting** [13, 16]. More precisely, using plain gradient-updates for encoding new evidence into an existing network will interfere with the existing knowledge, since this knowledge is encoded in a distributed manner across all the network's parameters $\boldsymbol{w}$ and not protected from being overwritten.

For these reasons, many algorithms have been developed that attempt to overcome forgetting. However, to avoid trivial solutions, a problem definition for CL needs to respect the notion of knowledge transfer [101]. In particular, we want to improve performance on previously learned skills by seeing new evidence (called **backward transfer**) while being able to learn new concepts faster or more in a data-efficient way (i. e., exploiting existing knowledge, called **forward transfer**) [102]. Simply speaking, we are interested in algorithms for training neural networks continually without suffering from forgetting while facilitating knowledge transfer. The common scenario investigated in the literature is the consecutive learning of a sequence of tasks as we will outline in Sec. 4.1. However, in the author's opinion, the focus on this type of CL can be explained by the initially introduced benchmarks, which caused the development of new benchmarks that are consistent with old ones, thus largely restricting CL research to the notion of learning a series of distinct tasks. To potentially better address the intuitive notion of lifelong learning, we propose an alternative type of CL in Sec. 4.2, which might lead to more practical relevance of CL algorithms in the future.

4.1 CONTINUALLY LEARNING A SEQUENCE OF TASKS

In this section, we consider *task-focused CL*, i. e., CL problems with a sequence of tasks which should be learned one after the other with a single model. Each task is represented by a dataset $\mathcal{D}^{(t)}$ for $t = 1, \dots, T$ and is associated with its own data-generating process $\mathcal{D}^{(t)} \overset{\text{i.i.d.}}{\sim} p^{(t)}(\boldsymbol{x})p^{(t)}(\boldsymbol{y} \mid \boldsymbol{x})$. During learning, access is only granted to one dataset $\mathcal{D}^{(t)}$ at a time, without access to past or future data, and the goal of learning is to obtain a

1 Due to the focus on neural network research, we exclusively consider gradient-based optimization (cf. Sec. 2.2.1).

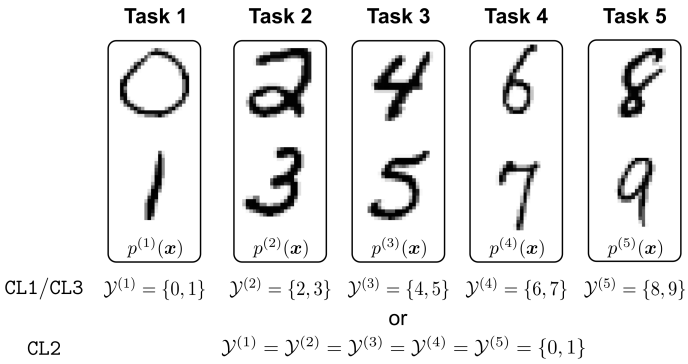


FIGURE 4.1: A typical CL benchmark called *SplitMNIST* [103]. The benchmark consists of 5 binary classification problems. Shown are two example inputs per task drawn from $p^{(t)}(\boldsymbol{x})$. Different CL scenarios consider different target spaces. In CL1 and CL3 the output space $\mathcal{Y} = \bigcup_t \mathcal{Y}^{(t)}$ has ten elements, while in CL2 it has only two $\mathcal{Y} = \{0, 1\}$. However, CL1 learners differ from CL3 by having access to the task identity during inference. See main text for details.

model that exhibits good performance on all tasks seen so far (we will be more specific below).

The literature on CL is vast and there are many variants of this framework. For instance, whether sample points in $\mathcal{D}^{(t)}$ can only be used once for forming weight updates (often referred to as *online* CL [99]) or whether tasks can be reoccurring. While all these considerations are important in practice, we attempt to form a clear and general overview over CL in this section, rather than striving to cover all variations found in the literature.

To gain a better understanding we start with an example benchmark and introduce some basic terminology that may prove useful later on. In Fig. 4.1 we illustrate the SplitMNIST benchmark, which is constructed from the popular MNIST dataset [30]. MNIST is a 10-way classification problem, containing 60K training examples of handwritten digits $0 - 9$. SplitMNIST is constructed by splitting this dataset into five binary classification problems as illustrated in Fig. 4.1. The overall goal to be achieved by learning these five tasks depends on the considered CL scenario. Most commonly, the following three scenarios are distinguished [104–106]: *task-incremental learning* (or **CL1**), *domain-incremental learning* (or **CL2**) and *class-incremental learning* (or **CL3**).

In CL1, the task-identity of an input $\boldsymbol{x}$ is always available during inference, and can be seen as an additional input to the model. In the case of

SplitMNIST, this means that five consecutive binary decision problems have to be learned (e. g., each with the class labels $\{0, 1\}$), but the availability of the task identity allows assigning each input $\boldsymbol{x}$ to one out of ten classes. Thus, the model maps to the output space $\mathcal{Y} = \{0, \dots, 9\}$. How the conditioning on the task-identity is implemented may vary in practice [e. g., 33], but is most often performed via a so-called *multi-head* setup. In this case, the neural network has a separate output layer (or head) per task. Therefore, the network's parameters are split into shared and task-specific parameters [40], and continual learning only has to occur in the shared body (i. e., preventing forgetting and facilitating transfer). This CL scenario is often considered the simplest one. Indeed, CL algorithms usually achieve best performance when being tested under `CL1`. However, especially in recurrent neural networks, the problem of catastrophic interference is still prominent for modern CL approaches even under `CL1` evaluation. Here, the requirement of maintaining a working memory in the neural activity causes a particular fragility with respect to parameter changes as we study in Ref. [33]. Yet, for feed-forward networks, the other two CL scenarios are meanwhile of broader interest.

In `CL2` and `CL3` the task-identity is not provided as an input to the model during inference. In `CL2` the output spaces among tasks are assumed to be identical: $\mathcal{Y}^{(s)} = \mathcal{Y}^{(t)}$ for $s \neq t$. Thus, in the case of SplitMNIST, the overall output space is $\mathcal{Y} = \{0, 1\}$, and hence the final model still performs a binary classification but on all 10 digits ($0, 2, 4, 6, 8$ vs. $1, 3, 5, 7, 9$ or even vs. odd). In general, task switches in `CL2` simply refer to changes in the input distribution (tasks cover different "input domains"). A more intuitive example might be the following: Let the problem to be solved be the distinction between handwritten digits 0 and 1, and let each task represent examples obtained from a different human. After having seen all tasks, we want to be able to distinguish handwritten zeros and ones, but we do not seek to identify the human (task-identity) that drew the current input. `CL2` is often called *single head* (as opposed to *multi-head*), as all tasks use the same output layer. In our opinion, and as illustrated with the SplitMNIST example, common CL benchmarks do not make intuitive sense when applied to `CL2`, which is why `CL2` is often just considered as an intermediate difficulty level between `CL1` and `CL3`.[2] However, `CL2` could also be more naturally understood as being able to incorporate new data for improving existing skills, an aspect we later discuss.

2 Converting a 10-way classification dataset to a binary decision problem in such ad hoc manner seems odd.

Lastly, CL3 or *class-incremental learning*[3] allows the output space $\mathcal{Y}$ to grow with new tasks, but without requiring access to task-identities during inference (in contrast to CL1). Thus, in the case of SplitMNIST, a CL3 learner is able to perform a full 10-way classification after having seen data from five binary classification problems. Here, the output layer is sometimes referred to as *growing-head* as new neurons need to be added to the output layer to accommodate new classes. This makes CL3 particularly challenging in classification as networks usually use a normalization over output units (e. g., a softmax). Thus, output weights for previously observed classes might need to be drastically re-tuned if the number of neurons over which the normalization is performed changes.[4] However, there are ways to define a likelihood function for classification without requiring an explicit normalisation [e. g., 107, 108]. Finally, we should note that when classes are allowed to reoccur across different tasks, then CL3 encompasses CL2. In particular, new evidence can be incorporated to improve existing skills while new skills (in the form of new classes) can be learnt.

In summary, the three CL scenarios for SplitMNIST are as follows:

- CL1: Given task-identity, discriminate: $\{0\}$ vs. $\{1\}$ or $\{2\}$ vs. $\{3\}$ or $\{4\}$ vs. $\{5\}$ or $\{6\}$ vs. $\{7\}$ or $\{8\}$ vs. $\{9\}$
- CL2: Discriminate: $\{0, 2, 4, 6, 8\}$ vs. $\{1, 3, 5, 7, 9\}$
- CL3: Discriminate all vs. all: $\{0\}$ vs. $\{1\}$ vs. $\{2\}$ vs. $\{3\}$ vs. $\{4\}$ vs. $\{5\}$ vs. $\{6\}$ vs. $\{7\}$ vs. $\{8\}$ vs. $\{9\}$

In practice, these distinctions are useful and we will see in Chapters 5 and 6 that CL methods exhibit considerate performance differences depending on the CL setting. However, it is crucial to note that the three learning scenarios above differ in the problem being solved by the "final" model. For instance, in CL3-SplitMNIST the final problem is MNIST (digit recognition), in CL2-SplitMNIST it is odd vs. even and in CL1-SplitMNIST it is a binary classification upon receiving a tuple consisting of an image and a task-identifier as input. In the rest of this discussion, we thus move away from the CL1/CL2/CL3 distinction and focus instead on finding CL algorithms based on the "final" problem $p(\boldsymbol{y} \mid \boldsymbol{x})$ to be solved.

Let's consider that the final problem of interest is approximating $p(\boldsymbol{y} \mid \boldsymbol{x})$ using a discriminative model $p(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{w})$. This could be, for instance, the

3 The name already indicates that most CL benchmarks are classification problems.

4 Starting with a huge output layer with many unused units does not trivially solve the problem, as the softmax pushes the weights of unused units to negative infinity [40].

problem of classifying MNIST digits as belonging to one of the ten possible classes. According to Def. 2, we assume the existence of a data-generating process $p(\boldsymbol{x})p(\boldsymbol{y} \mid \boldsymbol{x})$ from which the dataset is sampled (i.e., MNIST). In (task-focused) CL, however, we do not have direct access to an i.i.d. dataset from $p(\boldsymbol{x})p(\boldsymbol{y} \mid \boldsymbol{x})$ for learning, and instead assume that the data-generating process is split into tasks as follows:

$$p(\boldsymbol{x})p(\boldsymbol{y} \mid \boldsymbol{x}) = \sum_{t=1}^{T} \gamma_t p^{(t)}(\boldsymbol{x})p^{(t)}(\boldsymbol{y} \mid \boldsymbol{x}) \qquad \sum_{t=1}^{T} \gamma_t = 1 \,;\, \gamma_t \geq 0 \tag{4.1}$$

where learning has to occur sequentially on datasets $\mathcal{D}^{(t)} \overset{\text{i.i.d.}}{\sim} p^{(t)}(\boldsymbol{x})p^{(t)}(\boldsymbol{y} \mid \boldsymbol{x})$. The weighting factor γ_t can in the simplest case be assumed to be uniform $\gamma_t = 1/T$. Using this decomposition, learning a discriminative model $p(\boldsymbol{y} \mid \boldsymbol{w}, \boldsymbol{x})$ amounts to approximating:

$$p(\boldsymbol{y} \mid \boldsymbol{x}) = \frac{\sum_{t=1}^{T} \gamma_t p^{(t)}(\boldsymbol{x})p^{(t)}(\boldsymbol{y} \mid \boldsymbol{x})}{p(\boldsymbol{x})} = \sum_{t=1}^{T} \frac{\gamma_t p^{(t)}(\boldsymbol{x})}{p(\boldsymbol{x})} p^{(t)}(\boldsymbol{y} \mid \boldsymbol{x}) \tag{4.2}$$

There are several strategies that can be followed for finding a suitable $\boldsymbol{w}$. The standard nomenclature for CL algorithms is a division into three categories [96]: (i) *replay-based CL*, where stored or generated data from previous tasks is replayed during learning, (ii) *regularization-based CL*, where weight-updates are constrained for protecting existing knowledge, and (iii) *dynamic architectures*, where the network architecture dynamically grows to account for the storage of new knowledge.

In the following subsections, we will present our own nomenclature of CL algorithms which is directly focused on different ways to approach Eq. 4.2. We will summarize those approaches in Sec. 4.1.5.

Remark 2 *Sometimes, learning from a non-stationary distribution [22] is also referred to as continual learning [e.g., 109, 110]. A typical example is a changing input distribution due to sensor degradation. In such cases, the focus is not on overcoming forgetting but on quick adaption and thus transfer learning. Therefore, it should not be confused with the continual learning studied in this thesis, where some unknown but stationary $p(\boldsymbol{x})p(\boldsymbol{y} \mid \boldsymbol{x})$ is assumed, but in contrast to Def. 2 no access to an i.i.d. dataset is granted.*

4.1.1 *Replay-based CL*

Replay-based approaches train the model $p(\boldsymbol{y} \mid \boldsymbol{w}, \boldsymbol{x})$ on current data $\mathcal{D}^{(t)}$ mixed with replayed data $\tilde{\mathcal{D}}^{(s)}_{\text{replay}}$ from previous tasks $s < t$. Thus, the overall data available for training at task t is $\tilde{\mathcal{D}}^{(1)}_{\text{replay}} \cup \cdots \cup \tilde{\mathcal{D}}^{(t-1)}_{\text{replay}} \cup \mathcal{D}^{(t)}$.

There are two major types of replay that have to be distinguished, based on whether methods store or generate replay data. In the former case, the replayed datasets are stored subsets of the original data $\tilde{\mathcal{D}}^{(s)}_{\text{replay}} \subseteq \mathcal{D}^{(s)}$. This approach is often called *experience replay* [111] and many methods have been explored for selecting a *coreset* of data points to be stored [e. g., 112, 113]. These coresets $\tilde{\mathcal{D}}^{(s)}_{\text{replay}}$ are typically much smaller than the original datasets, which causes an imbalance when naively computing gradient updates, and requires a proper rescaling of loss terms (e. g., see Sec. 5 in [113]). Furthermore, as a heuristic to prevent overfitting on the coreset data, often only input data $\boldsymbol{x}$ from the coresets is used in combination with targets computed via a checkpointed model $p(\boldsymbol{y} \mid \boldsymbol{w}^{(t-1)}, \boldsymbol{x})$. Here, $\boldsymbol{w}^{(t-1)}$ denotes the network parameters checkpointed after completing training on the previous task. The goal is that while training $\boldsymbol{w}$ on $\mathcal{D}^{(t)}$, the predictive distributions for inputs from $\tilde{\mathcal{D}}^{(s)}_{\text{replay}}$ for $s < t$ stay close to the ones that were obtained with $p(\boldsymbol{y} \mid \boldsymbol{w}^{(t-1)}, \boldsymbol{x})$ (e. g., by using techniques such as *knowledge distillation* [114]).

Another option is so-called pseudo-replay or generative replay. Here, generative models (cf. Sec. 2.1.2) $p(\boldsymbol{x} \mid \boldsymbol{v}^{(s)})$ that approximate $p^{(s)}(\boldsymbol{x})$ are learned via $\mathcal{D}^{(s)}$ while learning on task s. When learning task $t > s$, replay data can be obtained via the stored generative model $\tilde{\mathcal{D}}^{(s)}_{\text{replay}} \sim p(\boldsymbol{x} \mid \boldsymbol{v}^{(s)})$. To avoid having a separate generative model per task, traditional approaches continually retrain a single generative model $p(\boldsymbol{x} \mid \boldsymbol{v}^{(1:s)})$ on a mixture of current data $\mathcal{D}^{(s)}$ and replayed data from $p(\boldsymbol{x} \mid \boldsymbol{v}^{(1:s-1)})$ [e. g., 104, 115]. During learning of task $t > s$, data replayed from $p(\boldsymbol{x} \mid \boldsymbol{v}^{(1:t-1)})$ can be labelled via the checkpointed model $p(\boldsymbol{y} \mid \boldsymbol{w}^{(t-1)}, \boldsymbol{x})$. However, as we demonstrate in Ref. [39], this approach leads to a recursive amplification of approximation errors for long task sequences, which is why we propose storing task-specific parameters $\boldsymbol{v}^{(s)}$ of the generative model $p(\boldsymbol{x} \mid \boldsymbol{v}^{(s)})$ via a shared hypernetwork (cf. Chapter 5). Interestingly, there are also approaches that replay internal representations from the neural network (i. e., neural activities) rather than just input data [e. g., 106].

It is worth pointing out that in generative replay, the problem of continual learning is essentially shifted to the generative model, while the discriminator is trained on pseudo-data from all tasks. In Ref. [116], we propose a novel idea that allows generative replay via the uncertainty landscape of the posterior predictive distribution $p(\boldsymbol{y} \mid \mathcal{D}, \boldsymbol{x})$, and thus without the need of an external generative model (cf. Chapter 7). However, all these approaches rely on explicit generative modelling in order to learn a discriminative model. For this reason, the performance of such CL algorithms heavily depends on the complexity of the input data $\boldsymbol{x}$, and performance declines rapidly for more difficult benchmarks [33, 106].

We finish our exposition on replay-methods with a remark on empirical performance. Currently, experience replay methods as described above perform substantially worse than a model trained from scratch only on coresets $\bigcup_{t=1}^{T} \tilde{\mathcal{D}}^{(t)}_{\text{replay}}$ [e. g., see 99]. Note, that training a model only on coresets is equivalent to normal i.i.d. training (no CL) but with a much smaller dataset than $\mathcal{D} \equiv \bigcup_{t=1}^{T} \mathcal{D}^{(t)}$. By contrast, the model trained continually via experience replay had access to all data from $\mathcal{D}$, but not in an i.i.d. manner. Thus, we can draw two conclusions from this result. First, a continually trained model via experience replay can exhibit worse performance than one trained from scratch on the stored coresets, indicating that our current way of incorporating coresets into weight updates is harmful. Second, empirical findings in current ML research do not support the hypothesis that the hippocampus uses experience replay to facilitate continual learning in the neocortex (this hypothesis is a simple instantiation of the complementary learning systems theory [15, 16, 96]).

4.1.2 *The Bayesian recursive update*

We introduced the Bayesian approach to learning in Chapter 3 in the context of uncertainty estimation. In this section, we discover another fundamental advantage of such probabilistic treatment. In particular, Bayes' rule can be applied recursively to incorporate new data, and thus continual learning is naturally included in the Bayesian framework. In the context of task-focused CL, the posterior of task t can be formed by using the posterior of the previous task $p(\boldsymbol{w} \mid \mathcal{D}^{(1:t-1)})$ as prior in Bayes' rule:

$$p(\boldsymbol{w} \mid \mathcal{D}^{(1:t)}) = \frac{p(\mathcal{D}^{(t)} \mid \boldsymbol{w}) p(\mathcal{D}^{(1:t-1)} \mid \boldsymbol{w}) p(\boldsymbol{w})}{p(\mathcal{D}^{(1:t)})} \tag{4.3}$$

$$= \frac{p(\mathcal{D}^{(t)} \mid \boldsymbol{w}) p(\boldsymbol{w} \mid \mathcal{D}^{(1:t-1)})}{p(\mathcal{D}^{(t)} \mid \mathcal{D}^{(1:t-1)})} \tag{4.4}$$

We used the conditional independence of the data in the above derivation (cf. Fig. 3.4). The conditional $p(\mathcal{D}^{(t)} \mid \mathcal{D}^{(1:t-1)})$ is constant with respect to $\boldsymbol{w}$ and we can write the recursive Bayesian update in this simplified form:

$$p(\boldsymbol{w} \mid \mathcal{D}^{(1:t)}) \propto p(\mathcal{D}^{(t)} \mid \boldsymbol{w}) p(\boldsymbol{w} \mid \mathcal{D}^{(1:t-1)}) \tag{4.5}$$

As mentioned in Sec. 3.5, Bayesian inference in neural networks is intractable and approximations are necessary. Let $q_{\theta^{(1:t)}}(\boldsymbol{w})$ be the approximate posterior $p(\boldsymbol{w} \mid \mathcal{D}^{(1:t)})$ after having learned t tasks. It thus appears natural to facilitate continual learning via the same recursive rule:

$$q_{\theta^{(1:t)}}(\boldsymbol{w}) \propto p(\mathcal{D}^{(t)} \mid \boldsymbol{w}) q_{\theta^{(1:t-1)}}(\boldsymbol{w}) \tag{4.6}$$

Note, that only the current data $\mathcal{D}^{(t)}$ enters Eq. 4.6, and all previously acquired knowledge enters via the "prior" $q_{\theta^{(1:t-1)}}(\boldsymbol{w})$. Thus, prevention of forgetting is essentially facilitated via the prior, which is why Farquhar & Gal [117] termed CL methods based on Eq. 4.6 as **prior-focused**. We will comment in Sec. 4.1.2.1 on the pitfalls of this naive substitution of real posterior with approximate posteriors.

Note that prior-focused methods are instantiations of Bayesian CL methods, which are inspired by the Bayesian recursive update. However, while Eq. 4.5 naturally implies CL, Bayesian-inspired CL methods can also be attained via different means (cf. Chapter 6). For instance, replay-methods have a Bayesian interpretation by combating forgetting via the likelihood as opposed to the prior [117]. Therefore, prior-focused methods should only be seen as a subclass of Bayesian CL.

In Sec. 3.5.1 and Sec. 3.5.2 we introduced variational inference (VI) and the Laplace approximation to perform approximate inference with neural networks. Both of these methods have been used to construct prior-focused CL methods [e. g., 84, 85, 102, 118–121]. We will quickly outline how these two approximate inference methods can be instantiated for CL.

The arguably most well known prior-focused CL adaption of the Laplace approximation (Sec. 3.5.2) is *elastic weight consolidation* [EWC 84, 102, 118].[5]

5 The original EWC algorithm from Kirkpatrick *et al.* [118] deviates mathematically from its Bayesian motivation (see the note by Huszár [84]). This was later corrected by Schwarz *et al.*

Here, the posterior of the previous task is approximated by a Gaussian $q_{\theta^{(1:t-1)}}(\boldsymbol{w})$ whose parameters are comprised in θ. To apply prior-focused learning, we consider $q_{\theta^{(1:t-1)}}(\boldsymbol{w})$ as prior with $q_{\theta^{(1:0)}}(\boldsymbol{w}) \equiv p(\boldsymbol{w})$ and form the Laplace approximation of task t by first finding the MAP:

$$\boldsymbol{w}^{(1:t)} = \arg\min_{\boldsymbol{w}} \underbrace{-\log p(\mathcal{D}^{(t)} \mid \boldsymbol{w})}_{\text{NLL}} \underbrace{-\log q_{\theta^{(1:t-1)}}(\boldsymbol{w})}_{\text{CL regularizer}} \tag{4.7}$$

We will come back to the precise form of the regularizer below. After learning $\boldsymbol{w}^{(1:t)}$, we can consider a second-order Taylor expansion to form a Gaussian approximation (cf. Sec. 3.5.2). In particular, the negative precision matrix is taken as the sum of Hessians $H_{\log p(\mathcal{D}^{(t)}|\boldsymbol{w})} + H_{\log q_{\theta^{(1:t-1)}}(\boldsymbol{w})}$, and $H_{\log p(\mathcal{D}^{(t)}|\boldsymbol{w})}$ is further approximated by the empirical Fisher matrix $-N^{(t)}F^{\text{emp},(t)}$. If we assume the prior to be $p(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}; 0, \sigma^2_{\text{prior}} I)$ and carry out the complete recursion we recover the following precision matrix for $q_{\theta^{(1:t)}}(\boldsymbol{w})$:

$$\left(\Sigma^{(1:t)}\right)^{-1} = \frac{1}{\sigma^2_{\text{prior}}} I + \sum_{s=1}^{t} N^{(s)} F^{\text{emp},(s)} \tag{4.8}$$

In EWC, $F^{\text{emp},(t)}$ is further assumed to be diagonal.[6] Thus, as a sum of diagonal matrices, the approximate posterior has a diagonal covariance matrix. The regularization term in Eq. 4.7 thus becomes a simple quadratic regularization term:

$$\text{const.} + \frac{1}{2} \sum_{p=1}^{\dim(\boldsymbol{w})} \underbrace{\left(\frac{1}{\sigma^2_{\text{prior}}} + \sum_{s=1}^{t-1} N^{(s)} F^{\text{emp},(s)}_{p,p} \right)}_{:=\varrho_p^{(1:t-1)}} (w_p - w_p^{(1:t-1)})^2 \tag{4.9}$$

where $\boldsymbol{w}^{(1:0)}$ corresponds to the prior's mean ($\boldsymbol{w}^{(1:0)} \equiv 0$). Note, that the precision $\varrho_p^{(1:t-1)}$ can be seen as an *importance weight*, determining how rigid individual parameters w_p are. Note that even though this regularization is inspired by Eq. 4.5, the number of approximations that entered its derivation

[102] and termed *online EWC*. For brevity, we refer to online EWC simply as EWC in this thesis.

6 Note, that it is infeasible to maintain the full quadratic matrix $F^{\text{emp},(t)}$ for modern network architectures, where $\boldsymbol{w}$ has usually millions (or even hundreds of billions [97]) of parameters. A block-diagonal approximation has been investigated by Ritter, Botev & Barber [85].

may call for an interpretation of Eq. 4.9 as a heuristic approach to CL (e. g., a similar heuristic is proposed by Zenke, Poole & Ganguli [103]). Such heuristic interpretation of Eq. 4.9 suggests that weak precision values facilitate the integration of new knowledge and strong precision values protect previously acquired knowledge. The trade-off between these two desiderata is in general referred to as the *plasticity-stability dilemma* [96]. To allow more control over this trade-off, a regularization strength β is introduced in practice:

$$-\log p(\mathcal{D}^{(t)} \mid \boldsymbol{w}) + \frac{\beta}{2} \sum_{p=1}^{\dim(\boldsymbol{w})} \varrho_p^{(1:t-1)} (w_p - w_p^{(1:t-1)})^2 \tag{4.10}$$

In summary, EWC uses optimization to obtain $\boldsymbol{w}^{(1:t)}$ which requires maximum likelihood training with an added quadratic penalty to mitigate forgetting. After obtaining $\boldsymbol{w}^{(1:t)}$, the diagonal elements of the (empirical) Fisher are computed for updating precision elements $\varrho_p^{(1:t)}$. In total, EWC's CL regularizer requires the storing of $2\dim(\boldsymbol{w})$ elements, i. e., the previous posterior's mean $\boldsymbol{w}^{(1:t-1)}$ and precision vector $\varrho^{(1:t-1)}$. The derivation of this algorithm when using a mixture of task-specific and shared parameters (e. g., `CL1`) can be found in the SM of our work in Ref. [40].

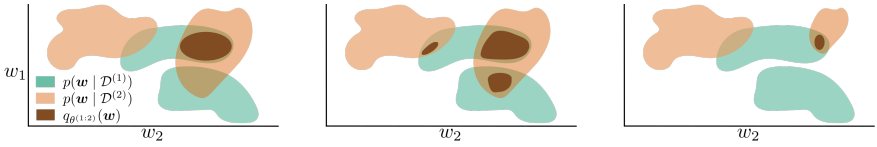
A second group of CL methods inspired by the recursive Bayesian update (Eq. 4.6) use variational inference (Sec. 3.5.1). In this case, the negative ELBO has to be minimized:

$$\theta^{(1:t)} = \arg\min_{\theta} \mathbb{E}_{q_\theta(\boldsymbol{w})} \left[-\log p(\mathcal{D} \mid \boldsymbol{w})\right] + \mathrm{KL}\left(q_\theta(\boldsymbol{w}); q_{\theta^{(1:t-1)}}(\boldsymbol{w})\right) \tag{4.11}$$

An instantiation of this idea for a variational family of Gaussian distributions with diagonal covariance matrix has been proposed by Nguyen *et al.* [119].[7] In particular, they adapted an algorithm, named *Bayes-by-Backprop* [BbB 77], to the CL setting. Since both the currently learned posterior approximation and the prior are Gaussian, the KL term (the so-called *prior-matching term*) can be evaluated analytically. The prior-matching can again be seen as a regularization to prevent forgetting. The reparametrization trick is used to allow a gradient-based optimization of an MC estimate of the expected NLL:

$$\mathbb{E}_{q_\theta(\boldsymbol{w})} \left[\log p(\mathcal{D} \mid \boldsymbol{w})\right] = \mathbb{E}_{\mathcal{N}(\boldsymbol{z};0,I)} \left[p(\mathcal{D} \mid \boldsymbol{w} = \mu + \sigma \odot \boldsymbol{z})\right] \tag{4.12}$$

7 Nguyen *et al.* [119] actually combined this idea with a mathematical sound incorporation of coresets, which we ignore here for the sake of presentation.



(a) unimodal (b) multimodal (c) task similarity

FIGURE 4.2: High density regions of the first and second task's posterior. The overlap regions illustrate admissible solutions for both tasks $p(\boldsymbol{w} \mid \mathcal{D}^{(1:2)})$, and the approximate posterior $q_{\theta^{(1:2)}}(\boldsymbol{w})$ obtained via a prior-focused method should be contained in these overlap regions. The family of distributions considered for finding $q_{\theta^{(1:2)}}(\boldsymbol{w})$ is one of the factors that influence CL performance. If $q_{\theta^{(1:2)}}(\boldsymbol{w})$ is unimodal **(a)**, as in EWC, only one overlap region is covered and an upcoming third tasks might not share solutions with this one. Multimodal approximations **(b)** are, on the other hand, difficult to scale (cf. Chapter 6). Another factor that influences CL performance is task similarity **(c)**, as smaller regions of overlap shrink the prior $q_{\theta^{(1:2)}}(\boldsymbol{w})$ of an upcoming third task.

where $\theta = \{\mu, \sigma\}$.

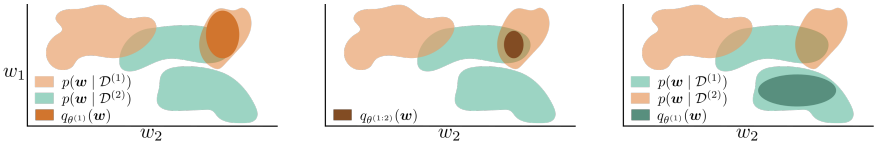
Nguyen *et al.* [119] call this algorithm *variational continual learning* (VCL). In Chapter 6, we will see examples of using variational inference for CL with more expressive variational families.

Interestingly, VCL is theoretically much more flexible than prior-focused learning based on the Laplace approximation due to the flexibility in choosing the variational family. However, due to increased optimization complexity (or increased difficulty in estimating the ELBO), simpler variational families are empirically preferable for complex problems [40]. Moreover, Eq. 4.11 is in practice more difficult to optimize than Eq. 4.10, which explains why a simple method such as EWC remains a compelling choice for prior-focused learning [33, 40].

4.1.2.1 *Recursive amplification of approximation errors*

(Thanks to Maria R. Cervera with whom the content of this section was developed in joint discussions.)

The Bayesian recursive update (Eq. 4.5) illustrates how the Bayesian framework allows a mathematical sound integration of new evidence. This is a striking observation, as the posterior $p(\boldsymbol{w} \mid \mathcal{D}^{(1:t)})$ is invariant to the order in which the data is observed. More specifically, under exact Bayesian inference, it does not matter whether we see all data at once or whether we see it continually. However, prior-focused methods often perform poorly in practice compared to other CL methods (at least in

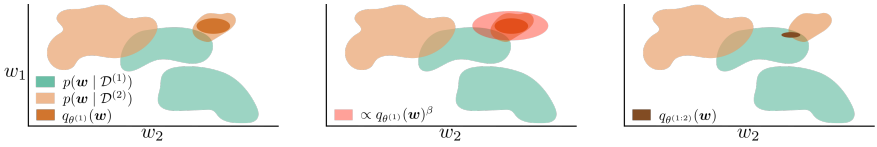


(a) 1st ordering (b) 1st ordering (c) 2nd ordering

FIGURE 4.3: Task order influences the solutions found with prior-focused methods. In this illustration, all high density regions of the orange task have an overlap with the green task but not vice versa. Thus, if the orange task is seen first **(a)**, a region of overlap can be found within the prior $q_{\theta^{(1)}}(\boldsymbol{w})$ **(b)**. However, if the green task is seen first **(c)**, there might be no admissible solutions for the orange task contained in the prior $q_{\theta^{(1)}}(\boldsymbol{w})$.

challenging scenarios) [40]. How can this discrepancy be understood given the theoretical optimality for knowledge integration via Eq. 4.5? The reason lies in the ad-hoc replacement of true posteriors with approximate posteriors when jumping from Eq. 4.5 to Eq. 4.6. The recursive nature of Eq. 4.6 causes a recursive amplification of approximation errors as we will intuitively illustrate in this section.

There is a multitude of factors that contribute to the poor performance often observed with prior-focused methods as we illustrate in Figs. 4.2, 4.3 and 4.4. If we would train i.i.d. on all tasks (no CL), e. g., via the optimization criterion in Eq. 3.5, we could easily identify solutions that work well on all tasks, ignoring those that only perform well on a subset of tasks. However, during CL (as studied in this section) we do not know the nature of upcoming tasks, and therefore have to always consider all solutions that perform well on tasks seen so far as being relevant for future ones. By its nature, approximate inference is (almost surely) erroneous, and thus potentially not capturing all solutions for seen tasks. For instance, consider Fig. 4.2a. Here, the chosen family of distributions for approximation are Gaussians with diagonal covariance matrix as illustrated by the axis-aligned ellipse (e. g., EWC). The posteriors of the first two tasks have multiple regions of overlap (multiple regions of weight configurations that perform well on both tasks). The approximate posterior $q_{\theta^{(1:2)}}(\boldsymbol{w})$, however, only covers one of these regions (Fig. 4.2b). The posterior of an upcoming third task $p(\boldsymbol{w} \mid \mathcal{D}^{(3)})$ might overlap with $p(\boldsymbol{w} \mid \mathcal{D}^{(1:2)})$ but not with $q_{\theta^{(1:2)}}(\boldsymbol{w})$. As a consequence, the third task could not be learned. By contrast, if all data would have been available at once, it would have been rather easy to form an approximate posterior that accommodates all three tasks. Thus, while the Bayesian re-



(a) untempered prior (b) tempered prior (c) forgetting

FIGURE 4.4: Tempering is a common strategy to increase plasticity of prior-focused methods but may lead to *graceful forgetting* [e. g., 40, 121]. The prior $q_{\theta^{(1)}}(\boldsymbol{w})$ in **(a)** has no region of overlap with the upcoming task. Tempering this prior via the inverse temperature β can create overlap regions with task 2 **(b)**. However, this artificial broadening of the prior causes non-admissible solutions for task 1 to be considered admissible. As a consequence, the approximate posterior $q_{\theta^{(1:2)}}(\boldsymbol{w})$ formed via the tempered prior $\propto q_{\theta^{(1)}}(\boldsymbol{w})^{\beta}$ is not fully contained in an orange region **(c)**. Thus, $q_{\theta^{(1:2)}}(\boldsymbol{w})$ suffers from forgetting as sampled solutions may not perform well on previous tasks.

cursive update (Eq. 4.5) does not distinguish between simultaneous and continual learning, applying approximate inference to the simultaneous or continual learning setting may lead to vastly different results. Better methods for approximate inference may mitigate (but not abolish) such effects (Fig. 4.2b), however, current empirical evidence suggests that theoretically more expressive approximations suffer disproportionately from scalability (cf. Chapter 6).

There are many other interplaying factors, such as task similarity (Fig. 4.2c) or task ordering (Fig. 4.3). As a result, heuristics (such as tempering, Fig. 4.4) have to be put in place to ensure an acceptable trade-off between stability and plasticity (cf. Eq. 4.10). The combination of these factors explains why prior-focused methods suffer in practice from rigidity and/or forgetting.

In summary, there is an unfortunate discrepancy between empirical results obtained from prior-focused methods and theoretical expectations cast by the Bayesian recursive update. This discrepancy can be intuitively understood as a result of the recursive amplification of errors arising due to the use of approximate inference. Therewith, prior-focused methods present a strong case for being cautious about the use of approximate inference, as predictions can vary drastically compared to performing inference via the true posterior. Interestingly, the fact that prior-focused methods often perform worse than other CL methods, that are not derived from Eq. 4.5, indicates that current ML research does not align with the simple hypothesis

that the brain integrates new knowledge by approximating the Bayesian recursive update [5].

4.1.3 *Task-specific solutions and explicit task-inference*

So far, we have discussed replay-based methods and prior-focused methods for tackling CL. In this section, we take a closer look at Eq. 4.2 and make a further assumption to derive algorithms that empirically often excel compared to the first two classes of algorithms that we encountered so far.

We start by noting that task-specific discriminative models $p^{(t)}(\boldsymbol{y} \mid \boldsymbol{x})$ can be learned in the usual manner (cf. Chapter 2) using the i.i.d. datasets $\mathcal{D}^{(t)}$. Thus, to evaluate Eq. 4.2, only the seemingly challenging factor $\frac{\gamma_t p^{(t)}(\boldsymbol{x})}{p(\boldsymbol{x})}$ needs to be estimated.

To overcome this hurdle, we introduce another assumption. Namely, we assume that the support of input distributions for distinct tasks are disjoint:

$$\text{supp}\{p^{(s)}(\boldsymbol{x})\} \cap \text{supp}\{p^{(t)}(\boldsymbol{x})\} = \emptyset \qquad \text{for} \qquad s \neq t \tag{4.13}$$

This is a strong assumption, but one that appears reasonable for common CL benchmarks (e. g., such as SplitMNIST; Fig. 4.1).

Under this assumption, the factor $\frac{p^{(t)}(\boldsymbol{x})}{p(\boldsymbol{x})}$ evaluates to 1 if $\boldsymbol{x} \in \text{supp}\{p^{(t)}(\boldsymbol{x})\}$ and 0 otherwise, which induces the problem of **task-inference** for an unseen input $\boldsymbol{x}$.

Task-inference means that, in order to make predictions via a task-specific model $p^{(t)}(\boldsymbol{y} \mid \boldsymbol{x})$, one first has to determine the task-identity t of an input $\boldsymbol{x}$. If a `CL1` scenario is considered, task-identity is provided and no task-inference is necessary, so continual learning is solved by learning task-specific models $p^{(t)}(\boldsymbol{y} \mid \boldsymbol{x})$ while preventing forgetting and facilitating transfer (e. g., by using progressive neural networks [122]).

Note, that the problem of task-inference is a classification problem, which has to be solved in a continual manner. However, task-inference is arguably a simpler discriminative problem than directly approximating $p(\boldsymbol{y} \mid \boldsymbol{x})$ according to Eq. 4.2. Thus, the CL approach considered in this section is a hierarchical approach with two stages, where the first stage is task identification and the second one is the problem of continually learning task-specific models $p^{(t)}(\boldsymbol{y} \mid \boldsymbol{x})$ where each can be learned via an i.i.d. dataset.

These two stages can be considered somewhat independent and thus different solutions can be considered for either of them. For instance, continually learning task-specific $p^{(t)}(\boldsymbol{y} \mid \boldsymbol{x})$ can be achieved by training separate

neural network modules (no interference with previous knowledge) which are laterally connected to previous ones to facilitate forward transfer [122]. In Chapters 5 and 6 we will use hypernetworks to realize a conceptually similar approach without the need of a growing architecture.

Similarly, task-inference can be approached in many ways [e. g., 39, 40, 123]. We will discuss replay- and uncertainty-based task-inference (as well as a combination of these two) in Chapters 5 and 6. To give a simple example, replay-based task-inference can be realized by training a separate classifier for task-identification continually using replay-techniques from Sec. 4.1.1.

To sum up, the CL approach presented in this section relies on an additional assumption (Eq. 4.13), namely that task-identity can be unequivocally inferred from inputs. This assumption allows to disentangle the CL problem into two subproblems, each of which is arguably simpler than the one induced by Eq. 4.2. More specifically, continual learning of task-specific models $p^{(t)}(\boldsymbol{y} \mid \boldsymbol{x})$ can be viewed as a `CL1` problem. Task-identification, on the other hand, can be viewed as a `CL3` problem, but for an overall simpler discriminative problem than the target $p(\boldsymbol{y} \mid \boldsymbol{x})$. Combining these two stages can be seen as solving an overall `CL3` (or `CL2`) problem (cf. Chapter 5).

4.1.4 *CL via conditional generative models*

In Chapter 3, we saw that Bayes' rule is the foundation for developing Bayesian statistics, which we utilized to develop CL algorithms in Sec. 4.1.2. In this section, we see another useful application of Bayes' rule for deriving CL algorithms.

Recently, the concept of *generative classifiers* was introduced to continual learning by van de Ven, Li & Tolias [124], which consider a classification problem $p(y \mid \boldsymbol{x})$ in which one class at a time is continually learned. We first review this idea from van de Ven, Li & Tolias [124] before applying it in general to task-focused CL.

Let $\mathcal{Y} = \{1, \ldots, K\}$. We can generally write the classification $p(y \mid \boldsymbol{x})$ as follows by using Bayes' rule:

$$p(y \mid \boldsymbol{x}) = \frac{p(\boldsymbol{x} \mid y)p(y)}{\sum_{y' \in \mathcal{Y}} p(\boldsymbol{x} \mid y')p(y')} \tag{4.14}$$

where $p(\boldsymbol{x} \mid y)$ is a class-conditioned generative model. We assume that all data corresponding to a class is presented at once. Therefore, we can

learn a separate generative model per class $p(\boldsymbol{x} \mid \boldsymbol{v}^{(y)})$ with parameters $\boldsymbol{v}^{(y)}$ without the need for a continual learning algorithm. In addition, we can assume $p(y)$ to be uniform, which results in:

$$p(y \mid \boldsymbol{x}) = \frac{p(\boldsymbol{x} \mid \boldsymbol{v}^{(y)})}{\sum_{y' \in \mathcal{Y}} p(\boldsymbol{x} \mid \boldsymbol{v}^{(y')})} \tag{4.15}$$

This idea can be generalized to tackle Eq. 4.2. Indeed, what we present in the following can be viewed as a generalization of the approach proposed by Lee *et al.* [125] (e. g., see Eq. 7 in [125]).

Consider learning task-specific generative models $p(\boldsymbol{x} \mid \boldsymbol{v}^{(t)})$ to capture $p^{(t)}(\boldsymbol{x})$ alongside with task-specific discriminative models to approximate $p^{(t)}(\boldsymbol{y} \mid \boldsymbol{x})$. This allows us to estimate the challenging factor $\frac{\gamma_t p^{(t)}(\boldsymbol{x})}{p(\boldsymbol{x})}$ from Eq. 4.2 by assuming $\gamma_t = \frac{N^{(t)}}{N}$ or simply $\gamma_t = 1/T$. In particular, we can approximate Eq. 4.2 as:

$$p(\boldsymbol{y} \mid \boldsymbol{x}) = \frac{\sum_{t=1}^{T} \gamma_t p^{(t)}(\boldsymbol{x}) p^{(t)}(\boldsymbol{y} \mid \boldsymbol{x})}{p(\boldsymbol{x})} \tag{4.16}$$

$$\approx \frac{\sum_{t=1}^{T} \frac{N^{(t)}}{N} p(\boldsymbol{x} \mid \boldsymbol{v}^{(t)}) p(\boldsymbol{y} \mid \boldsymbol{w}^{(t)}, \boldsymbol{x})}{\sum_{t=1}^{T} \frac{N^{(t)}}{N} p(\boldsymbol{x} \mid \boldsymbol{v}^{(t)})} \tag{4.17}$$

Here, we make the simplifying assumption that task-specific models have separate parametrizations, i. e., $\boldsymbol{v}^{(t)}$ and $\boldsymbol{w}^{(t)}$. In practice, however, generative and/or discriminative models can be employed that use shared as well as task-specific components. For instance, in Chapter 5 we will see how hypernetworks can be used to represent task-specific generative or discriminative models via a single shared meta-model.

Importantly, the generative models learned via this approach must be able to provide explicit density estimates. In comparison, the generative models required for generative replay in Sec. 4.1.1 only need to be able to provide sample points. Explicit density estimation on real-world tasks, such as natural images, is a challenging problem [126]. While the results obtained by van de Ven, Li & Tolias [124] and Lee *et al.* [125] are promising, further empirical validation is necessary to determine the scalability of this approach.

4.1.5 *Summary on task-focused CL*

We have categorized continual learning approaches in four different types, which are (i) replay-based methods (Sec. 4.1.1), (ii) prior-focused methods (Sec. 4.1.2), (iii) task-specific discriminators with explicit task-inference (Sec. 4.1.3) and (iv) task-specific discriminators and generators (Sec. 4.1.4). Not every CL method can be exclusively assigned to one of these categories. For instance, context-dependent gating [127] uses task-specific binary masks to select subnetworks and thus clearly belongs to category (iii). However, as these random subnetworks can be overlapping, they additionally use a regularization approach (e. g., EWC) to mitigate forgetting, such that the overall approach can be seen as a mixture of methods from categories (ii) and (iii). Furthermore, our categorization might not be seen as complete. For instance, even though heuristically motivated regularization approaches such as synaptic intelligence [103] and memory aware synapses [128] are somewhat theoretically related to EWC [129], it is unclear whether it is justified to see them as prior-focused methods, and thus as incarnations of the Bayesian recursive update. Despite these technical nuances, we consider our categorization as a constructive way of presenting the current state of research in continual learning.

Note, that for presenting categories (iii) and (iv) we strongly relied on the notion of tasks. The Bayesian recursive update, on the other hand, does technically not rely on the existence of well-defined tasks; an important insight that we will revisit in Sec. 4.2. However, most prior-focused methods are constructed in a way that tasks are necessary for computational feasibility. Also replay-methods do not strictly depend on the existence of tasks. Since in generative replay a generative model needs to be trained, access to (pseudo) i.i.d. input data is necessary (cf. Sec. 4.1.1) and thus the notion of tasks is beneficial (cf. Chapter 5). Coreset methods, on the other hand, are somewhat special and, to the best of our knowledge, not well understood. As we have seen in Sec. 4.1.1, they are outperformed by trivial baselines that have access to much less data but form proper i.i.d. updates [99]. Recall that using i.i.d. data allows, for instance via the NLL (Eq. 2.6), to form an unbiased consistent estimate of the gradient to minimize a divergence from the model to the ground-truth. By contrast, it is unclear what is optimized by a gradient estimate that is formed by using data from the current task plus a small stored subset of previously seen data points. Therefore, we currently see coreset methods as heuristically-motivated approach to CL,

and expect future research to provide a more formal understanding of their inner workings.

Lastly, we would like to stress again that category (iii) required an additional assumption (Eq. 4.13), which is, however, reasonable for common CL benchmarks. As CL algorithms from this category are the ones generally performing best (cf. Chapter 6), we hypothesize that the concentration on a more precisely defined subproblem is beneficial. Yet, CL algorithms do not appear to be practically relevant yet and we therefore argue that the problem of CL has to be split into more clearly defined subproblems, which allow the development of specialized but high-performing (and thus relevant) algorithms. In the next section, we will outline an example of such subproblem.

4.2 LEARNING FROM AUTOCORRELATED SAMPLE POINTS

(Thanks to Maria R. Cervera and Alexander Meulemans for constructive feedback on this section.)

In Sec. 4.1, we discussed task-focused CL, which is the type of CL most commonly studied in the literature as reflected by standard benchmarks (such as SplitMNIST, Fig. 4.1). However, is it also the most natural type of CL to study? To approach this question, let's reconsider the introductory example of Chapter 2, i. e., steering angle prediction. Here, the dataset $\mathcal{D}$ is generated by recording a human driver such that consecutive sample points are auto-correlated. Once a sufficient amount of driving experience was recorded, the dataset can be shuffled and approximately considered an i.i.d. sample. Thus, a model $p(\boldsymbol{y} \mid \boldsymbol{w}, \boldsymbol{x})$ can be trained on $\mathcal{D}$ using the statistical learning framework from Chapter 2. If the model needs to be improved, more data has to be collected and mixed with $\mathcal{D}$ for retraining.

This process seems unnatural. Instead, we would like to formulate the learning problem in a way, that the feedforward model $p(\boldsymbol{y} \mid \boldsymbol{w}, \boldsymbol{x})$ is trained in an online manner while receiving autocorrelated sample points just like the human driver.

Experience replay is in principle applicable to this scenario and found successful adaptations in the past for removing autocorrelations [e. g., 130]. But as we discussed in the previous section, experience replay (or coresets) does not excel in continual learning (at least not for moderate sizes of the replay buffer).

Also, the Bayesian recursive update (Eq. 4.5) is applicable to this scenario, and we could recursively approximate the posterior over all data seen so far for each incoming data point [e. g., 131]. However, the recursive

application of approximate inference leads to poor performance in practice (cf. Sec. 4.1.2.1), and applying this recursion to every incoming data point rather than to individual tasks can only worsen this effect.

The other CL approaches discussed in Sec. 4.1 are not applicable, since there is no apparent notion of tasks.

Note, that both experience replay and the Bayesian recursive update are agnostic to the specific type of autocorrelations being observed. More precisely, while we are not aware of a mathematical justification of experience replay (cf. Sec. 4.1.5), the Bayesian recursive update is completely generic and applicable to essentially any type of data stream.

However, for the problem at hand, we do not require such powerful algorithm as the incoming data stream of front-camera images follows very specific dynamics. To make this idea more concrete, let's reconsider, as an example, MCMC methods as discussed in Sec. 3.5.3 but applied to inputs $\boldsymbol{x}$ rather than to parameters $\boldsymbol{w}$. In particular, let's consider a chain $\pi(\boldsymbol{x}^{(t)} \mid \boldsymbol{x}^{(t-1)})$ constructed such that the stationary distribution is the distribution of front-camera images $p(\boldsymbol{x})$. As discussed in Sec. 3.5.3, consecutive sample points $\boldsymbol{x}^{(t-1)}, \boldsymbol{x}^{(t)}, \boldsymbol{x}^{(t+1)}$ are autocorrelated. Assuming $\pi(\boldsymbol{x}^{(t)} \mid \boldsymbol{x}^{(t-1)})$ is fixed and given (or learned separately), can we use it to construct a π-specific CL algorithm for learning a feedforward model $p(\boldsymbol{y} \mid \boldsymbol{w}, \boldsymbol{x})$ from tuples $\ldots, (\boldsymbol{x}^{(t-1)}, \boldsymbol{y}^{(t-1)}), (\boldsymbol{x}^{(t)}, \boldsymbol{y}^{(t)}), \ldots$ with autocorrelated inputs according to $\pi(\boldsymbol{x}^{(t)} \mid \boldsymbol{x}^{(t-1)})$?

Let τ be a trajectory of length T sampled via $\pi(\boldsymbol{x}^{(t)} \mid \boldsymbol{x}^{(t-1)})$, i. e., $\tau[t] = \boldsymbol{x}^{(t)}$ and $\boldsymbol{x}^{(t)} \sim \pi(\boldsymbol{x} \mid \tau[t-1])$. In conventional supervised learning the goal is to minimize the *true risk*, for instance, by using the negative log-likelihood as cost function as discussed in Chapter 2:

$$\mathcal{R} \coloneqq \mathbb{E}_{p(\boldsymbol{x},\boldsymbol{y})}[-\log p(\boldsymbol{y} \mid \boldsymbol{w}, \boldsymbol{x})] \tag{4.18}$$

$$= -\int_{\boldsymbol{x}} p(\boldsymbol{x}) \mathbb{E}_{p(\boldsymbol{y}|\boldsymbol{x})}[\log p(\boldsymbol{y} \mid \boldsymbol{w}, \boldsymbol{x})] \, d\boldsymbol{x} \tag{4.19}$$

The true risk can be approximated via Monte-Carlo using and i.i.d. dataset $\mathcal{D} \overset{\text{i.i.d.}}{\sim} p(\boldsymbol{x}, \boldsymbol{y})$, which leads to the so-called *empirical risk*

$$\mathcal{R}^{\text{emp}} \coloneqq -\frac{1}{N} \sum_{n=1}^{N} \log p(\boldsymbol{y}_n \mid \boldsymbol{w}, \boldsymbol{x}_n) \tag{4.20}$$

Moreover, gradients of the empirical risk are unbiased estimates of the gradient of the true risk and are consistent with respect to a growing dataset size.

In our setting, however, it might be constructive to consider another quantity that represents the true risk, that we call *evolving risk*

$$\mathcal{R}_{\pi}^{\text{evolve}} := \lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}_{p(\boldsymbol{y}|\tau[t])}[-\log p(\boldsymbol{y} \mid \boldsymbol{w}, \tau[t])] \to \mathcal{R} \tag{4.21}$$

where we use the fact that $p(\boldsymbol{x})$ is the stationary distribution of the chain.

Thus, minimizing $\mathcal{R}_{\pi}^{\text{evolve}}$ is equivalent to minimizing the true risk $\mathcal{R}$. However, to accomplish the goal of this section, gradient estimates would need to be formed by approximating a local version of the evolving risk:

$$\mathcal{R}_{\pi,\Delta t}^{\text{evolve}} := \frac{1}{\Delta t} \sum_{t=t_1}^{t_2} \mathbb{E}_{p(\boldsymbol{y}|\tau[t])}[-\log p(\boldsymbol{y} \mid \boldsymbol{w}, \tau[t])] \tag{4.22}$$

$$\approx -\frac{1}{\Delta t} \sum_{t=t_1}^{t_2} \log p(\boldsymbol{y}^{(t)} \mid \boldsymbol{w}, \tau[t]) \qquad \text{with } \boldsymbol{y}^{(t)} \overset{\text{i.i.d.}}{\sim} p(\boldsymbol{y} \mid \boldsymbol{x}) \tag{4.23}$$

with $\Delta t = t_2 - t_1 + 1$.

Note, that gradients from such estimate are still unbiased if $\Delta t = 1$ (mini-batch of size 1). However, the gradient noise cannot be described as white noise as it would be the case for stochastic gradient descent [132], and thus cannot be assumed to cancel out in expectation for consecutive gradient steps (at consecutive time points). Therefore, parameter updates should be informed about the dynamics $\pi(\boldsymbol{x}^{(t)} \mid \boldsymbol{x}^{(t-1)})$.

We unfortunately have no analytic solution to such problem yet, but consider it an important and interesting future direction. Note, that considering the dynamics $\pi(\boldsymbol{x}^{(t)} \mid \boldsymbol{x}^{(t-1)})$ as fixed is reasonable for real-world agents, and biological systems might have evolved learning rules that are optimized for learning from autocorrelated sample points. Moreover, even without an analytically derived update rule for learning in this setting, modern ML tools from the field of meta-learning [133] could be used to learn how to continually learn as already explored in other settings [134].

4.3 CLOSING REMARKS ON CONTINUAL LEARNING

We discussed the current state of CL research in this chapter, emphasizing that most research efforts are concentrated on *task-focused CL* (i. e., setups with a discrete set of tasks to be learned one after the other, cf. Sec. 4.1). This will also be the CL setting under consideration in Chapters 5 and 6. We

already mentioned, and will confirm later, that more specialized algorithms (e. g., specialized to task-focused CL) often perform better than generic alternatives (e. g., based on the Bayesian recursive update). Moreover, we have seen that many approaches to CL rely on explicit generative modelling or Bayesian inference. These considerations lead to the conclusion that the current CL literature focuses on a too generic problem, which does not allow for (computationally) simple and effective solutions. Therefore, we proposed a potential subcategory for continual learning research in Sec. 4.2, which is inspired by the way how biological agents learn continually within their environment, i. e., by learning from autocorrelated sample points. Focusing on more strictly constrained subproblems will hopefully lead to the development of algorithms that are effective and scalable enough to become relevant for real-world applications. In the end, there is a natural desire for continual learning as efficiency requires us to find ways of incorporating new evidence without retraining, and we hope that the contributions of this chapter can contribute in achieving this goal.

5

CONTINUAL LEARNING VIA HYPERNETWORKS

This chapter's content is taken from a publication in the Proceedings of the International Conference on Learning Representations (2020) and can be found online in an extended form. The original publication is ***authored by****: Johannes von Oswald*, Christian Henning*, João Sacramento* and Benjamin F. Grewe [39].*

* *These authors contributed equally.*

As discussed in Chapter 4, neural networks suffer from catastrophic forgetting when they are sequentially trained on multiple tasks. To overcome this problem, we present in this chapter an approach based on task-conditioned hypernetworks, i.e., networks that generate the weights of a target model based on task identity. Continual learning (CL) is less difficult for this class of models thanks to a simple key feature: instead of recalling the input-output relations of all previously seen data, task-conditioned hypernetworks only require rehearsing task-specific weight realizations, which can be maintained in memory using a simple regularizer. Besides achieving state-of-the-art performance on standard CL benchmarks, additional experiments on long task sequences reveal that task-conditioned hypernetworks display a very large capacity to retain previous memories. Notably, such long memory lifetimes are achieved in a compressive regime, when the number of trainable hypernetwork weights is comparable or smaller than target network size. We provide insight into the structure of low-dimensional task embedding spaces (the input space of the hypernetwork) and show that task-conditioned hypernetworks demonstrate transfer learning. Finally, forward information transfer is further supported by empirical results on a challenging CL benchmark based on the CIFAR-10/100 image datasets.

5.1 INTRODUCTION

We use the same notation as in the previous chapters where $f(\boldsymbol{x}; \boldsymbol{w})$ denotes a neural network and tasks are represented by a sequence of datasets $\mathcal{D}^{(1)}$, $\ldots$, $\mathcal{D}^{(T)}$, with $\|\mathcal{D}^{(1)}\| = N^{(t)}$.

Here, we propose a method that addresses the continual learning problem as explained in Sec. 4.1.3. In particular, we propose addressing catastrophic

forgetting at the meta level: instead of directly attempting to retain $f(\boldsymbol{x}; \boldsymbol{w})$ for previous tasks, we fix the outputs of a metamodel $h(\boldsymbol{e}; \psi)$ termed *task-conditioned hypernetwork* which maps a task embedding $\boldsymbol{e}$ to weights $\boldsymbol{w}$. Now, a *single* point has to be memorized per task. To motivate such approach, we perform a thought experiment: we assume that we are allowed to store all inputs $\{\boldsymbol{x}^{(t,n)} \mid t \in [1 \dots T], n \in [1 \dots N^{(t)}]\}$ seen so far, and to use these data to compute model outputs corresponding to $\boldsymbol{w}^{(T-1)}$. In this idealized setting, one can avoid forgetting by simply mixing data from the current task with data from the past, $\{(\boldsymbol{x}^{(s,n)}, \hat{\boldsymbol{y}}^{(s,n)}) \mid s \in [1 \dots T-1], n \in [1 \dots N^{(s)}]\} \cup \{(\boldsymbol{x}^{(T,n)}, \boldsymbol{y}^{(T,n)}) \mid n \in [1 \dots N^{(T)}]\}$, where $\hat{\boldsymbol{y}}^{(s,n)}$ refers to the synthetic targets generated using the model itself $f(\boldsymbol{x}^{(s,n)}, \boldsymbol{w}^{(T-1)})$. Hence, by training to retain previously acquired input-output mappings, one can obtain a sequential algorithm in principle as powerful as multi-task learning. Multi-task learning, where all tasks are learned simultaneously, can be seen as a CL upper-bound. The strategy described above has been termed rehearsal [135]. However, storing previous task data violates our CL desiderata.

Therefore, we introduce a change in perspective and move from the challenge of maintaining individual input-output data points to the problem of maintaining sets of parameters $\{\boldsymbol{w}^{(t)}\}$, without explicitly storing them. To achieve this, we train the metamodel parameters ψ analogous to the above outlined learning scheme, where synthetic targets now correspond to weight configurations that are suitable for previous tasks. This exchanges the storage of an entire dataset by a single low-dimensional task descriptor, yielding a massive memory saving in all but the simplest of tasks. Despite relying on regularization, our approach is a conceptual departure from previous algorithms based on regularization in weight [e.g., 103, 118] or activation space [e.g., 136].

Our experimental results show that task-conditioned hypernetworks do not suffer from catastrophic forgetting on a set of standard CL benchmarks. Remarkably, they are capable of retaining memories with practically no decrease in performance, when presented with very long sequences of tasks. Thanks to the expressive power of neural networks, task-conditioned hypernetworks exploit task-to-task similarities and transfer information forward in time to future tasks. Finally, the task-conditional metamodelling perspective that we put forth is generic, as it does not depend on the specifics of the target network architecture. We exploit this key principle and show that the very same metamodelling framework extends to, and can improve, an important class of CL methods known as generative replay

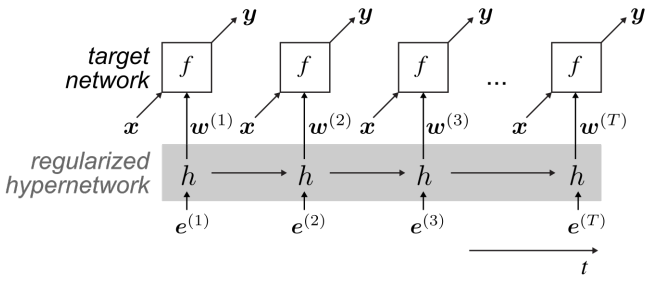


FIGURE 5.1: **Task-conditioned hypernetworks for continual learning.** Commonly, the parameters of a neural network are directly adjusted from data to solve a task. Here, a weight generator termed *hypernetwork* is learned instead. Hypernetworks map embedding vectors to weights, which parameterize a target neural network. In a continual learning scenario, a set of task-specific embeddings is learned via backpropagation. Embedding vectors provide task-dependent context and bias the hypernetwork to particular solutions.

methods, which are current state-of-the-art performers in many practical problems [104, 115, 137].

5.2 MODEL

5.2.1 *Task-conditioned hypernetworks*

HYPERNETWORKS PARAMETERIZE TARGET MODELS. The centerpiece of our approach to continual learning is the hypernetwork, Fig. 5.1. Instead of learning the parameters $\boldsymbol{w}$ of a particular function f directly (the *target model*), we learn the parameters ψ of a metamodel. The output of such metamodel, the hypernetwork, is $\boldsymbol{w}$. Hypernetworks can therefore be thought of as weight generators, which were originally introduced to dynamically parameterize models in a compressed form [35, 36, 138, 139].

CONTINUAL LEARNING WITH HYPERNETWORK OUTPUT REGULARIZATION. One approach to avoid catastrophic forgetting is to store data from previous tasks and corresponding model outputs, and then fix such outputs.

This can be achieved using an output regularizer of the following form, where past outputs play the role of pseudo-targets [135, 140, 141]:

$$\mathcal{L}_{\text{output}} = \sum_{t=1}^{T-1} \sum_{n=1}^{N^{(t)}} \|f(\boldsymbol{x}^{(t,n)}, \boldsymbol{w}^*) - f(\boldsymbol{x}^{(t,n)}, \boldsymbol{w})\|^2, \tag{5.1}$$

In the equation above, $\boldsymbol{w}^*$ is the set of parameters before attempting to learn task T, and f is the learner. This approach, however, requires storing and iterating over previous data, a process that is known as *rehearsing*. This is potentially expensive memory-wise and not strictly online learning. A possible workaround is to generate the pseudo-targets by evaluating f on random patterns [135] or on the current task dataset [140]. However, this does not necessarily fix the behavior of the function f in the regions of interest.

Hypernetworks sidestep this problem naturally. In target network weight space, a *single* point (i.e., one set of weights) has to be fixed per task. This can be efficiently achieved with task-conditioned hypernetworks, by fixing the hypernetwork output on the appropriate task embedding.

Similar to Benjamin, Rolnick & Kording [141], we use a two-step optimization procedure to introduce memory-preserving hypernetwork output constraints. First, we compute a candidate change $\Delta\psi$ which minimizes the current task loss $\mathcal{L}_{\text{task}}^{(T)} = \mathcal{L}_{\text{task}}(\psi, \boldsymbol{e}^{(T)}, \mathcal{D}^{(T)})$ with respect to ψ. The candidate $\Delta\psi$ is obtained with an optimizer of choice [we use Adam throughout; 142]. The actual parameter change is then computed by minimizing the following total loss:

$$\begin{aligned}\mathcal{L}_{\text{total}} &= \mathcal{L}_{\text{task}}(\psi, \boldsymbol{e}^{(T)}, \mathcal{D}^{(T)}) + \mathcal{L}_{\text{output}}(\psi^*, \psi, \Delta\psi, \{\boldsymbol{e}^{(t)}\}) &(5.2)\\ &= \mathcal{L}_{\text{task}}(\psi, \boldsymbol{e}^{(T)}, \mathcal{D}^{(T)}) + \frac{\beta}{T-1} \sum_{t=1}^{T-1} \|h(\boldsymbol{e}^{(t)}; \psi^*) - h(\boldsymbol{e}^{(t)}, \psi + \Delta\psi))\|^2 &(5.3)\end{aligned}$$

where ψ^* is the set of hypernetwork parameters before attempting to learn task T, $\Delta\psi$ is considered fixed and β is a hyperparameter that controls the strength of the regularizer. In SM D of Ref. [39], we run a sensitivity analysis on β and experiment with a more efficient stochastic regularizer where the averaging is performed over a random subset of past tasks.

More computationally-intensive algorithms that involve a full inner-loop refinement, or use second-order gradient information by backpropagating through $\Delta\psi$ could be applied. However, we found empirically that our one-step correction worked well. Exploratory hyperparameter scans revealed

that the inclusion of the lookahead $\Delta\psi$ in Eq. 5.3 brought a minor increase in performance, even when computed with a cheap one-step procedure. Note that unlike in Eq. 5.1, the memory-preserving term $\mathcal{L}_{\text{output}}$ does not depend on past data. Memory of previous tasks enters only through the collection of task embeddings $\{\boldsymbol{e}^{(t)}\}_{t=1}^{T-1}$.

LEARNED TASK EMBEDDINGS. Task embeddings are differentiable deterministic parameters that can be learned, just like ψ. At every learning step of our algorithm, we also update the current task embedding $\boldsymbol{e}^{(T)}$ to minimize the task loss $\mathcal{L}_{\text{task}}^{(T)}$. After learning the task, the final embedding is saved and added to the collection $\{\boldsymbol{e}^{(t)}\}$.

5.2.2 *Model compression with chunked hypernetworks*

CHUNKING. In a straightforward implementation, a hypernetwork produces the entire set of weights of a target neural network. For modern deep neural networks, this is a very high-dimensional output. However, hypernetworks can be invoked iteratively, filling in only part of the target model at each step, in *chunks* [35, 42]. This strategy allows applying smaller hypernetworks that are reusable. Interestingly, with chunked hypernetworks it is possible to solve tasks in a compressive regime, where the number of learned parameters (those of the hypernetwork) is effectively smaller than the number of target network parameters.

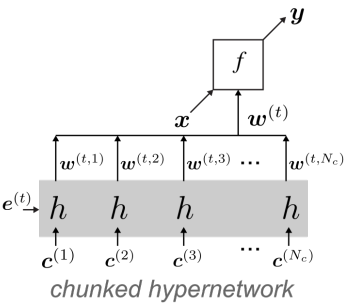


FIGURE 5.2: **Model compression.** A smaller, chunked hypernetwork can be used iteratively, producing a chunk of target network weights at a time (e.g., one layer at a time). Chunked hypernetworks can achieve model compression: the effective number of trainable parameters can be smaller than the number of target network weights.

CHUNK EMBEDDINGS AND NETWORK PARTITIONING. Reapplying the same hypernetwork multiple times introduces weight sharing across partitions of the target network, which is usually not desirable. To allow for a flexible parameterization of the target network, we introduce a set $\mathcal{C} = \{\boldsymbol{c}^{(i)}\}_{i=1}^{N_c}$ of chunk embeddings,

which are used as an additional input to the hypernetwork, Fig. 5.2. Thus, the full set of target network parameters $\boldsymbol{w} = [h(\boldsymbol{e}, \boldsymbol{c}^{(1)}), \ldots, h(\boldsymbol{e}, \boldsymbol{c}^{(N_c)})]$ is produced by iteration over $\mathcal{C}$, keeping the task embedding $\boldsymbol{e}$ fixed. This way, the hypernetwork can produce distinct weights for each chunk. Furthermore, chunk embeddings, just like task embeddings, are ordinary deterministic parameters that we learn via backpropagation. For simplicity, we use a shared set of chunk embeddings for all tasks and we do not explore special target network partitioning strategies.

How flexible is our approach? Chunked neural networks can in principle approximate any target weight configuration arbitrarily well. For completeness, we state this formally in SM E of Ref. [39].

5.2.3 *Context-free inference: unknown task identity*

DETERMINING WHICH TASK TO SOLVE FROM INPUT DATA. Our hypernetwork requires a task embedding input to generate target model weights. In certain CL applications, an appropriate embedding can be immediately selected as task identity is unambiguous, or can be readily inferred from contextual clues. In other cases, knowledge of the task at hand is not explicitly available during inference. In the following, we show that our metamodelling framework generalizes to such situations. In particular, we consider the problem of inferring which task to solve from a given input pattern, a noted benchmark challenge [105, 143]. Below, we explore two different strategies that leverage task-conditioned hypernetworks in this CL setting.

TASK-DEPENDENT PREDICTIVE UNCERTAINTY. Neural network models are increasingly reliable in signalling novelty and appropriately handling out-of-distribution data. For categorical target distributions, the network ideally produces a flat, high entropy output for unseen data and, conversely, a peaked, low-entropy response for in-distribution data [144, 145]. This suggests a first, simple method for task inference (**HNET+ENT**). Given an input pattern for which task identity is unknown, we pick the task embedding which yields lowest predictive uncertainty, as quantified by output distribution entropy. While this method relies on accurate novelty detection, which is in itself a far from solved research problem, it is otherwise straightforward to implement and no additional learning or model is required to infer task identity.

HYPERNETWORK-PROTECTED SYNTHETIC REPLAY. When a generative model is available, catastrophic forgetting can be circumvented by mixing current task data with replayed past synthetic data [for recent work see 115, 137]. Besides protecting the generative model itself, synthetic data can protect another model of interest, for example, another discriminative model as we discussed in Sec. 4.1.1. This conceptually simple strategy is in practice often performing best if the input data is easy to model, e. g., on MNIST-derived benchmarks [105]. Inspired by these successes, we explore augmenting our system with a replay network, here a standard variational autoencoder [VAE; 79] (but see SM F in Ref. [39] for experiments with a generative adversarial network [146].

Synthetic replay is a strong, but not perfect, CL mechanism as the generative model is subject to drift, and errors tend to accumulate and amplify with time. Here, we build upon the following key observation: just like the target network, the generator of the replay model can be specified by a hypernetwork. This allows protecting it with the output regularizer, Eq. 5.3, rather than with the model's own replay data, as done in related work. Thus, in this combined approach, both synthetic replay and task-conditional metamodelling act in tandem to reduce forgetting.

We explore hypernetwork-protected replay in two distinct setups. First, we consider a minimalist architecture (**HNET+R**), where *only the replay model*, and not the target classifier, is parameterized by a hypernetwork. Here, forgetting in the target network is obviated by mixing current data with synthetic data. Synthetic target output values for previous tasks are generated using a soft targets method, i.e., by simply evaluating the target function before learning the new task on synthetic input data. Second (**HNET+TIR**), we introduce an auxiliary task inference classifier, protected using synthetic replay data and trained to predict task identity from input patterns. This architecture requires additional modelling, but it is likely to work well when tasks are strongly dissimilar. Furthermore, the task inference subsystem can be readily applied to process more general forms of contextual information, beyond the current input pattern. We provide additional details, including network architectures and the loss functions that are optimized, in SM B and SM C of Ref. [39].

5.3 RESULTS

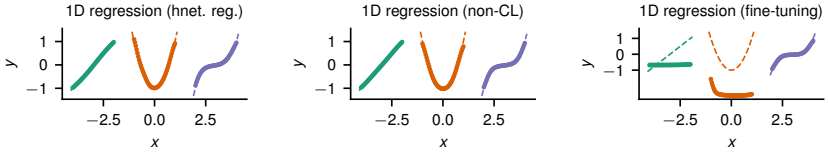
We evaluate our method on a set of standard image classification benchmarks on the MNIST, CIFAR-10 and CIFAR-100 public datasets[1]. Our main aims are to (1) study the memory retention capabilities of task-conditioned hypernetworks across three continual learning settings, and (2) investigate information transfer across tasks that are learned sequentially.

CONTINUAL LEARNING SCENARIOS. In our experiments we consider the three different CL scenarios from van de Ven & Tolias [105] that we introduced in Sec. 4.1. To briefly reiterate them: In `CL1`, the task identity is given to the system. This is arguably the standard sequential learning scenario (at least at the time of publication), and the one we consider unless noted otherwise. In `CL2`, task identity is unknown to the system, but it does not need to be explicitly determined. A target network with a fixed head is required to solve multiple tasks. In `CL3`, task identity has to be explicitly inferred. It has been argued that this scenario is the most natural, and the one that tends to be harder for neural networks [105, 143].

EXPERIMENTAL DETAILS. Aiming at comparability, for the experiments on the MNIST dataset we model the target network as a fully-connected network and set all hyperparameters after van de Ven & Tolias [105], who recently reviewed and compared a large set of CL algorithms. For our CIFAR experiments, we opt for a ResNet-32 target neural network [147] to assess the scalability of our method. A summary description of the architectures and particular hyperparameter choices, as well as additional experimental details, is provided in SM C of Ref. [39]. We emphasize that, on all our experiments, the number of hypernetwork parameters is always smaller or equal than the number of parameters of the models we compare with.

NONLINEAR REGRESSION TOY PROBLEM. To illustrate our approach, we first consider a simple nonlinear regression problem, where the function to be approximated is scalar-valued, Fig. 5.3. Here, a sequence of polynomial functions of increasing degree has to be inferred from noisy data. This motivates the continual learning problem: when learning each task in succession by modifying ψ with the memory-preserving regularizer turned off ($\beta = 0$, see Eq. 5.3) the network learns the last task but forgets previ-

1 Source code is available under https://github.com/chrhenning/hypercl.



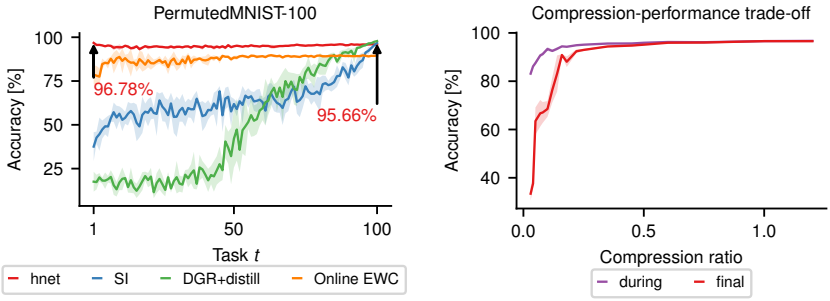
(a) (b) (c)

FIGURE 5.3: **1D nonlinear regression. (a)** Task-conditioned hypernetworks with output regularization can easily model a sequence of polynomials of increasing degree, while learning in a continual fashion. **(b)** The solution found by a target network which is trained directly on all tasks simultaneously is similar. **(c)** Fine-tuning, i.e., learning sequentially, leads to forgetting of past tasks. Dashed lines depict ground truth, markers show model predictions.

ous ones, Fig. 5.3c. The regularizer protects old solutions, Fig. 5.3a, and performance is comparable to an offline non-continual learner, Fig. 5.3b.

PERMUTED MNIST BENCHMARK. Next, we study the permuted MNIST benchmark [148]. This problem is set as follows. First, the learner is presented with the full MNIST dataset. Subsequently, novel tasks are obtained by applying a random permutation to the input image pixels. This process can be repeated to yield a long task sequence, with a typical length of $T = 10$ tasks. Given the low similarity of the generated tasks, permuted MNIST is well suited to study the memory capacity of a continual learner. For $T = 10$, we find that task-conditioned hypernetworks are state-of-the-art on `CL1`, Table 5.1. Interestingly, inferring tasks through the predictive distribution entropy (HNET+ENT) works well on the permuted MNIST benchmark. Despite the simplicity of the method, both synaptic intelligence [SI; 103] and online elastic weight consolidation [EWC; 102] are overperformed on `CL3` by a large margin (SI results are reported in Table 1 in Ref. [39]). When complemented with generative replay methods, task-conditioned hypernetworks (HNET+TIR and HNET+R) are the best performers on all three CL scenarios.

Performance differences become larger in the long sequence limit, Fig. 5.4a. For longer task sequences ($T = 100$), SI and DGR+distill [104, 115] degrade gracefully, while the regularization strength of online EWC prevents the method from achieving high accuracy (see Fig. A6 in Ref. [39] for a hyperparameter search on related work). Notably, task-conditioned hypernetworks show minimal memory decay and find high performance solutions. Be-



(a) (b)

FIGURE 5.4: **Experiments on the permuted MNIST benchmark. (a)** Final test set classification accuracy on the t-th task after learning one hundred permutations (PermutedMNIST-100). Task-conditioned hypernetworks (hnet, in red) achieve very large memory lifetimes on the permuted MNIST benchmark. Synaptic intelligence [SI, in blue; 103], online EWC [in orange; 102] and deep generative replay [DGR+distill, in green; 104, 115] methods are shown for comparison. Memory retention in SI and DGR+distill degrade gracefully, whereas EWC suffers from rigidity and can never reach very high accuracy, even though memories persist for the entire experiment duration. **(b)** Compression ratio $\frac{\dim(\psi)+\sum_t \dim(\boldsymbol{e}^{(t)})}{\dim(\boldsymbol{w})}$ versus task-averaged test set accuracy after learning all tasks (labelled 'final', in red) and immediately after learning a task (labelled 'during', in purple) for the PermutedMNIST-10 benchmark. Hypernetworks allow for model compression and perform well even when the number of target model parameters exceeds their own. Performance decays nonlinearly: accuracies stay approximately constant for a wide range of compression ratios below unity. Hyperparameters were tuned once for compression ratio ≈ 1 and were then used for all compression ratios. Shaded areas denote STD (a) resp. SEM (b) across 5 random seeds.

	EWC	DGR	HNET+ENT	HNET+TIR	HNET+R
P10-CL1	$95.96^{\pm 0.06}$	$97.51^{\pm 0.01}$	$97.57^{\pm 0.02}$	$97.57^{\pm 0.02}$	$\mathbf{97.87}^{\pm 0.01}$
P10-CL2	$94.42^{\pm 0.13}$	$97.35^{\pm 0.02}$	$92.80^{\pm 0.15}$	$97.58^{\pm 0.02}$	$\mathbf{97.60}^{\pm 0.01}$
P10-CL3	$33.88^{\pm 0.49}$	$96.38^{\pm 0.03}$	$91.75^{\pm 0.21}$	$97.59^{\pm 0.01}$	$\mathbf{97.76}^{\pm 0.01}$
S-CL1	$99.12^{\pm 0.11}$	$99.61^{\pm 0.02}$	$99.79^{\pm 0.01}$	$99.79^{\pm 0.01}$	$\mathbf{99.83}^{\pm 0.01}$
S-CL2	$64.32^{\pm 1.90}$	$96.83^{\pm 0.20}$	$87.01^{\pm 0.47}$	$94.43^{\pm 0.28}$	$\mathbf{98.00}^{\pm 0.03}$
S-CL3	$19.96^{\pm 0.07}$	$91.79^{\pm 0.32}$	$69.48^{\pm 0.80}$	$89.59^{\pm 0.59}$	$\mathbf{95.30}^{\pm 0.13}$

TABLE 5.1: Task-averaged test accuracy (± SEM, $n = 20$) on the permuted ('P10') and split ('S') MNIST experiments. In the table, EWC refers to online EWC and DGR refers to DGR+distill [results reproduced from 105]. We tested three hypernetwork-based models: for HNET+ENT (HNET alone for CL1), we inferred task identity based on the entropy of the predictive distribution; for HNET+TIR, we trained a hypernetwork-protected recognition-replay network (based on a VAE, cf. Fig. A1 in Ref. [39]) to infer the task from input patterns; for HNET+R the main classifier was trained by mixing current task data with synthetic data generated from a hypernetwork-protected VAE.

cause the hypernetwork operates in a compressive regime (see Fig. 5.4b and Fig. A7 in Ref. [39] for an exploration of compression ratios), our results do not naively rely on an increase in the number of parameters. Rather, they suggest that previous methods are not yet capable of making full use of target model capacity in a CL setting. We report a set of extended results on this benchmark in SM D of Ref. [39], including a study of CL2/3 ($T = 100$), where HNET+TIR strongly outperforms the related work.

SPLIT MNIST BENCHMARK. Split MNIST is another popular CL benchmark, designed to introduce task overlap (cf. Fig. 4.1). In this problem, the various digits are sequentially paired and used to form five binary classification tasks. Here, we find that task-conditioned hypernetworks are the best overall performers. In particular, HNET+R improves the previous state-of-the-art method DGR+distill on both CL2 and CL3, almost saturating the CL2 upper bound for replay models (see SM D in Ref. [39]). Since HNET+R is essentially hypernetwork-protected DGR, these results demonstrate the generality of task-conditioned hypernetworks as effective memory protectors. To further support this, in SM F in Ref. [39] we show that our replay models (we experiment with both a VAE and a GAN) can learn in

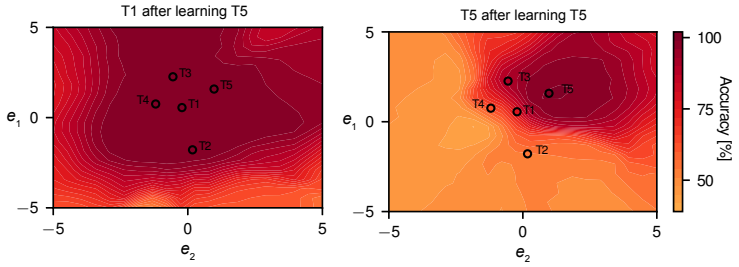


FIGURE 5.5: **Two-dimensional task embedding space for the split MNIST benchmark.** Color-coded test set classification accuracies after learning the five splits, shown as the embedding vector components are varied. Markers denote the position of final task embeddings. **(a)** High classification performance with virtually no forgetting is achieved even when e-space is low-dimensional. The model shows information transfer in embedding space: the first task is solved in a large volume that includes embeddings for subsequently learned tasks. **(b)** Competition in embedding space: the last task occupies a finite high performance region, with graceful degradation away from the embedding vector. Previously learned task embeddings still lead to moderate, above-chance performance.

a class-incremental manner the full MNIST dataset. Finally, HNET+ENT again outperforms EWC, without any generative modelling.

On the split MNIST problem, tasks overlap and therefore continual learners can transfer information across tasks. To analyze such effects, we study task-conditioned hypernetworks with two-dimensional task embedding spaces, which can be easily visualized. Despite learning happening continually, we find that the algorithm converges to a hypernetwork configuration that can produce target model parameters that simultaneously solve old and new tasks, Fig. 5.5, given the appropriate task embedding.

SPLIT CIFAR-10/100 BENCHMARK. Finally, we study a more challenging benchmark, where the learner is first asked to solve the full CIFAR-10 classification task and is then presented with sets of ten classes from the CIFAR-100 dataset. We perform experiments both with a high-performance ResNet-32 target network architecture (Fig. 5.6) and with a shallower model (see Fig. A3 in Ref. [39]) that we exactly reproduced from previous work [103]. Remarkably, on the ResNet-32 model, we find that task-conditioned hypernetworks essentially eliminate altogether forgetting. Furthermore, forward information transfer takes place; knowledge

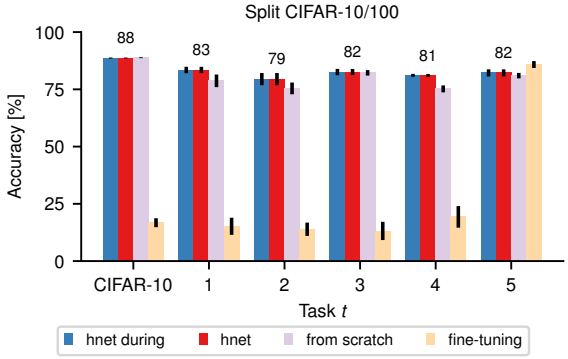


FIGURE 5.6: **Split CIFAR-10/100 CL benchmark.** Test set accuracies (mean $\pm$ STD, $n = 5$) on the entire CIFAR-10 dataset and subsequent CIFAR-100 splits of ten classes. Our hypernetwork-protected ResNet-32 displays virtually no forgetting; final averaged performance (hnet, in red) matches the immediate one (hnet-during, in blue). Furthermore, information is transferred across tasks, as performance is higher than when training each task from scratch (purple). Disabling our regularizer leads to strong forgetting (in yellow).

from previous tasks allows the network to find better solutions than when learning each task individually from initial conditions. Interestingly, forward transfer is stronger on the shallow model experiments, where we otherwise find that our method performs comparably to SI.

5.4 DISCUSSION

BAYESIAN ACCOUNTS OF CONTINUAL LEARNING. According to the standard Bayesian CL perspective, a posterior parameter distribution is recursively updated using Bayes' rule as tasks arrive [84, 118, 119]. While this approach is theoretically sound, in practice, the approximate inference methods that are typically preferred can lead to stiff models, as a compromise solution that suits all tasks has to be found within the mode determined by the first task. Such restriction does not apply to hypernetworks, which can in principle model complex multimodal distributions [41, 42, 44]. Thus, rich, hypernetwork-modelled priors are one avenue of improvement for Bayesian CL methods. Interestingly, *task-conditioning* offers an alternative possibility: instead of consolidating every task onto a single distribution, a shared task-conditioned hypernetwork could be leveraged to model a set

of parameter posterior distributions. This conditional metamodel naturally extends our framework to the Bayesian learning setting. Such approach will likely benefit from additional flexibility, compared to conventional recursive Bayesian updating. We will investigate this idea thoroughly in the next chapter.

RELATED APPROACHES THAT RELY ON TASK-CONDITIONING. Our model fits within, and in certain ways generalizes, previous CL methods that condition network computation on task descriptors. Task-conditioning is commonly implemented using multiplicative masks at the level of modules [122, 149], neurons [127, 150] or weights [151]. Such methods work best with large networks and come with a significant storage overhead, which typically scales with the number of tasks. Our approach differs by explicitly modelling the full parameter space using a metamodel, the hypernetwork. Thanks to this metamodel, generalization in parameter and task space is possible, and task-to-task dependencies can be exploited to efficiently represent solutions and transfer present knowledge to future problems. Interestingly, similar arguments have been drawn in work developed concurrently to ours [152], where task embedding spaces are further explored in the context of few-shot learning. In the same vein, and like the approach developed here, recent work in CL generates last-layer network parameters as part of a pipeline to avoid catastrophic forgetting [153] or distills parameters onto a contractive auto-encoding model [154].

POSITIVE BACKWARDS TRANSFER. In its current form, the hypernetwork output regularizer protects previously learned solutions from changing, such that only weak backwards transfer of information can occur. Given the role of selective forgetting and refinement of past memories in achieving intelligent behavior [155, 156], investigating and improving backwards transfer stands as an important direction for future research.

RELEVANCE TO SYSTEMS NEUROSCIENCE. Uncovering the mechanisms that support continual learning in both brains and artificial neural networks is a long-standing question [13, 96, 157]. We close with a speculative systems interpretation [158, 159] of our work as a model for modulatory top-down signals in cortex. Task embeddings can be seen as low-dimensional context switches, which determine the behavior of a modulatory system, the hypernetwork in our case. According to our model, the hypernetwork would in turn regulate the activity of a target cortical network.

As it stands, implementing a hypernetwork would entail dynamically changing the entire connectivity of a target network, or cortical area. Such a process seems difficult to conceive in the brain. However, this strict literal interpretation can be relaxed. For example, a hypernetwork can output lower-dimensional modulatory signals [160], instead of a full set of weights. This interpretation is consistent with a growing body of work which suggests the involvement of modulatory inputs in implementing context- or task-dependent network mode-switching [127, 161–163].

5.5 CONCLUSION

In this chapter, we introduced a novel neural network model, the task-conditioned hypernetwork, that is well-suited for the CL problems discussed in Sec. 4.1. A task-conditioned hypernetwork is a metamodel that learns to parameterize target functions, that are specified and identified in a compressed form using a task embedding vector. Past tasks are kept in memory using a hypernetwork output regularizer, which penalizes changes in previously found target weight configurations. This approach is scalable and generic, being applicable as a standalone CL method or in combination with generative replay. Our results are state-of-the-art on standard benchmarks (at the time of publication) and suggest that task-conditioned hypernetworks can achieve long memory lifetimes, as well as transfer information to future tasks, two essential properties of a continual learner.

However, the use of generative replay in methods HNET+R and HNET+TIR makes it cumbersome to scale these methods to complex real-world tasks. By contrast, the method HNET+ENT does not rely on generative modelling. We will therefore thoroughly explore this method in the next chapter and propose a probabilistic extension the here presented task-conditioned hypernetwork.

6

POSTERIOR-META-REPLAY FOR CONTINUAL LEARNING

This chapter's content is taken from a publication in the Proceedings of the Conference on Neural Information Processing Systems (2021) and can be found online in an extended form. The original publication is ***authored by****: Christian Henning*, Maria R. Cervera*, Francesco D'Angelo, Johannes von Oswald, Regina Traber, Benjamin Ehret, Seijin Kobayashi, Benjamin F. Grewe and João Sacramento [40].*

** These authors contributed equally.*

We have seen in Sec. 4.1.2, that Bayesian learning directly applies to continual learning, since recursive and one-off Bayesian updates yield the same result. In practice, however, recursive updating often leads to poor trade-off solutions across tasks because approximate inference is necessary for most models of interest (cf. Sec. 4.1.2.1). In this chapter, we describe an alternative Bayesian approach where task-conditioned parameter distributions are continually inferred from data. We offer a practical deep learning implementation of our framework based on probabilistic task-conditioned hypernetworks, an approach we term *posterior meta-replay*. Experiments on standard benchmarks show that our probabilistic hypernetworks compress sequences of posterior parameter distributions with virtually no forgetting. We obtain considerable performance gains compared to existing Bayesian CL methods, and identify task inference as our major limiting factor. This limitation has several causes that are independent of the considered sequential setting, opening up new avenues for progress in CL.

6.1 INTRODUCTION

As mentioned before, the advantages of a Bayesian approach for solving the problem of sequentially learning tasks $\mathcal{D}^{(1)}, \ldots, \mathcal{D}^{(T)}$ are numerous and include the ability to drop all i.i.d. assumptions across and within tasks in a mathematically sound way, the ability to revisit tasks whenever new data becomes available, and access to principled uncertainty estimates capturing both data and parameter uncertainty (cf. Chapter 3). The predominant Bayesian CL approach is based on the recursive Bayesian update and called **prior-focused** learning [117], as explained in Sec. 4.1.2. In theory, this

recursive update in Eq. 4.5 can always recover the posterior $p(\boldsymbol{w} \mid \mathcal{D}^{(1:T)})$, independently of how the data is presented. However, because proper Bayesian inference is intractable, approximations are needed in practice, which lead to errors that are recursively amplified. As a result, whether solutions that are easily found in the i.i.d. setting can be obtained via the approximate recursive update strongly depends on factors such as task ordering, task similarity and the considered family of distributions (see Sec. 4.1.2.1 for a discussion). These factors limit the effectiveness of the recursive update and have a detrimental effect on the performance of prior-focused methods, especially in CL settings where task-identity is not provided.

To overcome these limitations, we propose in this chapter an alternative Bayesian approach to CL that does not rely on the recursive update to learn distinct tasks and instead aims to learn task-specific posteriors (Fig. 6.1, refer to SM F.1 in Ref. [40] for a detailed discussion of the probabilistic graphical model). In this view, finding trade-off solutions across tasks is not required, and knowledge transfer can be explicitly controlled for each task via the prior, which is no longer prescribed by the recursive update and can thus be set freely. By introducing probabilistic extensions of the task-conditioned hypernetworks introduced in Chapter 5, we show how task-specific posteriors can be learned with a single shared meta-model, an approach we term **posterior meta-replay**.

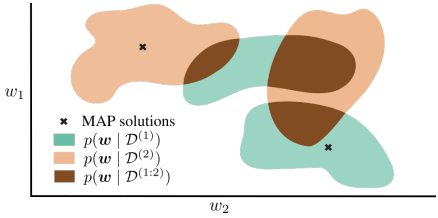


FIGURE 6.1: The proposed *posterior meta-replay* framework learns task-specific posteriors $p(\boldsymbol{w} \mid \mathcal{D}^{(t)})$ via a single shared meta-model, with task-specific point estimates (e.g., MAP) being a limit case. In this view, the modelled solution space is not limited to admissible solutions that lie in the overlap of all task-specific posteriors. By contrast, *prior-focused* methods learn a single posterior $p(\boldsymbol{w} \mid \mathcal{D}^{(1:T)})$ recursively and thus require the existence of trade-off solutions between learned and future tasks in the currently modelled solution space. Shaded areas indicate high density regions.

This approach introduces two challenges: forgetting at the level of the hypernetwork, and the need to know task identity to correctly condition the hypernetwork. We empirically show that forgetting at the meta-level can be prevented by using a simple regularizer that replays parameters of

previous posteriors (in analogy to the regularization performed in Chapter 5). In task-agnostic inference settings, i. e., CL3 or *class-incremental learning* as introduced in Sec. 4.1, the main hurdle therefore becomes task inference at test time. Here we focus on this task-agnostic setting, arguably the most challenging but also the most natural CL scenario, since the obtained models can be deployed just like those obtained via i.i.d. training (e.g., irrespective of the sequential training, the final model will be a classifier across all classes). In order to explicitly infer task identity from unseen inputs without resorting to generative models, we thoroughly study the use of principled uncertainty that naturally arises in Bayesian models (i. e., an extension and generalization of the HNET+ENT baseline introduced in Chapter 5). We show that results obtained in this task-agnostic setting with our approach constitute a leap in performance compared to prior-focused methods. Furthermore we show that limitations in task inference via predictive uncertainty are not related to our CL solution, but depend instead on the combination of approximate inference method, architecture, uncertainty measure and prior. Finally, we investigate how task inference can be further improved through several extensions.

We summarize our main contributions below:

- We describe a Bayesian CL framework where task-conditioned posterior parameter distributions are continually learned and compressed in a hypernetwork.
- In a series of synthetic and real-world CL benchmarks we show that our task-conditioned hypernetworks exhibit essentially no forgetting, both for explicitly parameterized and implicit posterior distributions, despite using the parameter budget of a single model.
- Compared to prior-focused methods, our approach leads to a leap in performance in task-agnostic inference while maintaining the theoretical benefits of a Bayesian approach.
- Our approach scales to modern architectures such as ResNets, and remaining performance limitations are linked to uncertainty-based out-of-distribution detection (cf. Chapter 7) but not to our CL solution.
- Finally, we show how prominent existing Bayesian CL methods such as elastic weight consolidation can be dramatically improved in task-agnostic settings by introducing a small set of task-specific parameters and explicitly inferring the task.

6.2 RELATED WORK

CONTINUAL LEARNING. Most related to the content of this chapter is our previous study (Ref. [39], presented in Chapter 5) that introduces task-conditioned hypernetworks for CL, and already considers task inference via predictive uncertainty in the deterministic case. Our framework can be seen as a probabilistic extension of the method from Chapter 5, which provides task-specific point estimates via a shared meta-model (cf. Sec. 6.3). Follow-up work also achieves task inference via predictive uncertainty, e.g., Wortsman *et al.* [164] use it to select a learned binary mask per task that modulates a random base network. Here we complement these studies by thoroughly exploring task inference via several uncertainty measures, disclosing the factors that limit task inference and highlighting the importance of parameter uncertainty.

A variety of methods tackling CL have been derived from a Bayesian perspective. A prominent example are *prior-focused* methods [117], which incorporate knowledge from past data via the prior and, in contrast to our work, aim to find a shared posterior for all data (cf. Sec. 4.1.2). Examples include (Online) EWC [102, 118] and VCL [119, 121]. Other methods like CN-DPM [125] use Bayes' rule for task inference on the joint $p(\boldsymbol{x}, c)$, where c is a discrete condition such as task identity (cf. Sec. 4.1.4). An evident downside of CN-DPM is the need for a separate generative and discriminative model per condition. More generally, such an approach requires meaningful density estimation in the input space, a requirement that is challenging for modern ML problems [126].

Other Bayesian CL approaches consider instead task-specific posterior parameter distributions. Lee *et al.* [165] learn separate task-specific Gaussian posterior approximations which are merged into a single posterior after all tasks have been seen. CBLN [166] also learns a separate Gaussian posterior approximation per task but later tries to merge similar posteriors in the induced Gaussian mixture model. Task inference is thus required and achieved via predictive uncertainty, although for a more reliable estimation all experiments consider batches of 200 samples that are assumed to belong to the same task. Tuor, Wang & Leung [167] also learn a separate approximate posterior per task and use predictive uncertainty for task-boundary detection and task inference. In contrast to these approaches, we learn all task-specific posteriors via a single shared meta-model and remain agnostic to the approximate inference method being used. A conceptually related approach is MERLIN [168], which learns task-specific weight distributions

by training an ensemble of models per task that is used as training set for a task-conditioned variational autoencoder. Importantly, MERLIN requires a fine-tuning stage at inference, such that every drawn model is fine-tuned on stored coresets, i.e., a small set of samples withheld throughout training. By contrast, our approach learns the parameters of an approximate Bayesian posterior $p(\boldsymbol{w} \mid \mathcal{D}^{(t)})$ per task t, and no fine-tuning of drawn models is required.

BAYESIAN NEURAL NETWORKS. Because neural networks are expressive enough to fit almost any data [169] and are often deployed in an over-parametrized regime, it is implausible to expect that any single solution obtained from limited data generalizes to the ground truth $p(\boldsymbol{y} \mid \boldsymbol{w}, \boldsymbol{x}) \approx p(\boldsymbol{y} \mid \boldsymbol{x})$ almost everywhere on $p(\boldsymbol{x})$. By contrast, Bayesian statistics (introduced in Chapter 3) considers a distribution over models, explicitly handling uncertainty to acknowledge data insufficiencies. How Bayesian statistics can can be applied to neural networks is discussed in Sec. 3.5.

While a deterministic discriminative model can only capture *aleatoric uncertainty*, a Bayesian treatment allows to also capture *epistemic uncertainty* by being uncertain about the model's parameters (*parameter uncertainty*, cf. Sec. 3.1). This proper treatment of uncertainty is of importance for safety-critical applications, where intelligent systems are expected to *know what they don't know* (but see Sec. 3.3). However, due to the complexity of modelling high-dimensional distributions at the scale of modern deep learning and due to the difficulties of encoding prior knowledge into the parameter space, BNNs still face severe scalability issues. Here, we employ several approximations to the posterior based on variational inference (cf. Sec. 3.5.1) from prior work, ranging from simple and scalable methods with a mean-field variational family like Bayes-by-Backprop (BbB, [77]) to methods with complex but rich variational families like the spectral Stein gradient estimator [82]. For more details see Sec. 6.3 and SM C in Ref. [40].

6.3 METHODS

In this section we describe our *posterior meta-replay* framework (Fig. 6.2). We start by introducing task-conditioned hypernetworks as a tool to continually learn parameters of task-specific posteriors, each of which is learned using variational inference (cf. Sec. 3.5.1 or SM C.1 in Ref. [40]). We then explain how the framework can be instantiated for both simple, explicit posterior approximations, and complex ones parametrized by an auxiliary network,

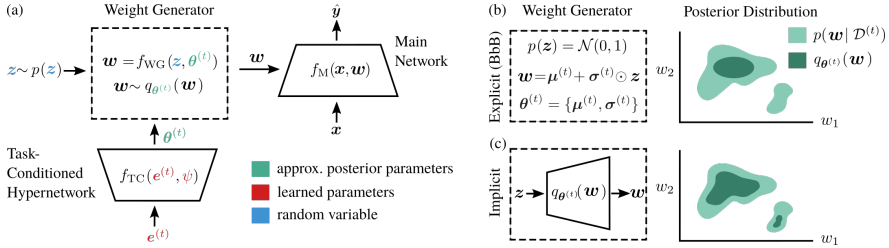


FIGURE 6.2: *Posterior meta-replay* for CL. **(a)** The architecture consists of a main network M that processes inputs $\boldsymbol{x}$ and generates predictions $\hat{\boldsymbol{y}}$ according to a set of weights $\boldsymbol{w}$ generated by a weight generator (WG). The WG is a deterministic function $f_{\text{WG}}(\boldsymbol{z};\boldsymbol{\theta}^{(t)})$ that transforms a base distribution $p(\boldsymbol{z})$ into a distribution over main network weights, where $\boldsymbol{\theta}^{(t)}$ are the parameters of the approximate posterior $q_{\boldsymbol{\theta}^{(t)}}(\boldsymbol{w})$. Crucially, $\boldsymbol{\theta}^{(t)}$ are task-specific, and generated by a task-conditioned (TC) hypernetwork, which receives task embeddings $\boldsymbol{e}^{(t)}$ as input. The embeddings and the parameters $\boldsymbol{\psi}$ of the TC are learned continually via a simple *meta-replay* regularizer (Eq. 6.1). **(b)** We refer to the approximate posteriors as *explicit* if f_{WG} is predefined. In Bayes-by-Backprop (BbB), for example, the reparametrization trick transforms Gaussian noise into weight samples. **(c)** More complex, *implicit* posterior approximations are parametrized by an auxiliary hypernetwork, which receives its task-conditioned parameters from the TC, which now plays the role of a hyper-hypernetwork. The obtained posterior approximations are more flexible and can, for example, capture multi-modality.

and describe how forgetting can be mitigated through the use of a meta-regularizer. We next explain how predictive uncertainty, naturally arising from a probabilistic view of learning, can be used to infer task identity for both **PosteriorReplay** methods, and **PriorFocused** methods that use a multihead output. Finally, we outline ways to boost task inference.

TASK-CONDITIONED HYPERNETWORKS. Traditionally, hypernetworks [35, 36] are seen as neural networks that generate the weights $\boldsymbol{w}$ of a main network **M** processing inputs as $\hat{\boldsymbol{y}} = f_{\text{M}}(\boldsymbol{x};\boldsymbol{w})$ (cf. Sec. 2.3). Here, we consider instead hypernetworks that learn to generate $\boldsymbol{\theta}$, the *parameters of a distribution* $q_{\boldsymbol{\theta}}(\boldsymbol{w})$ over main network weights. By taking low-dimensional task embeddings $\boldsymbol{e}^{(t)}$ as inputs and computing $\boldsymbol{\theta}^{(t)} = f_{\text{TC}}(\boldsymbol{e}^{(t)};\boldsymbol{\psi})$, task-conditional (**TC**) computation is possible. Sampling is realized by transforming a base distribution $p(\boldsymbol{z})$ via a weight generator (**WG**) $f_{\text{WG}}(\boldsymbol{z};\boldsymbol{\theta}^{(t)})$, whose choice determines the family of distributions considered for the approximation

(i.e., the *variational family*). In our framework, weights $\boldsymbol{w} \sim q_{\boldsymbol{\theta}^{(t)}}(\boldsymbol{w})$ are directly used for inference without requiring any fine-tuning.

Importantly, all learnable parameters are comprised in the TC system, which can be designed to have less parameters than the main network, i.e., $\dim(\psi) + \sum_t \dim(\boldsymbol{e}^{(t)}) < \dim(\boldsymbol{w})$. Such constraint is vital to ensure fairness when comparing different CL methods, and is enforced in all our computer vision experiments. Additional details can be found in SM C.2 in Ref. [40].

POSTERIOR-REPLAY WITH EXPLICIT DISTRIBUTIONS. Different families of distributions can be realized within our framework. In the special case of a point estimate $q_{\boldsymbol{\theta}^{(t)}}(\boldsymbol{w}) = \delta(\boldsymbol{w} - \boldsymbol{\theta}^{(t)})$, the WG system can be omitted altogether as it corresponds to the identity $\boldsymbol{\theta}^{(t)} = f_{\text{WG}}(\boldsymbol{z}; \boldsymbol{\theta}^{(t)})$. This reduces our solution to the deterministic CL method introduced in Chapter 5 (Ref. [39]), which we refer to as **PosteriorReplay-Dirac** in this chapter. However, capturing parameter uncertainty is a key ingredient of Bayesian statistics that is necessary for more robust task inference (cf. Sec. 6.4.2). We thus turn as a first step to *explicit* distributions $q_{\boldsymbol{\theta}^{(t)}}(\boldsymbol{w})$, for which the WG system samples according to a predefined function. We refer as **PosteriorReplay-Exp** to finding a mean-field Gaussian approximation via the BbB algorithm (see SM C.3.1 in Ref. [40]). In this case, $\boldsymbol{\theta}^{(t)}$ corresponds to the set of means and variances that define a Gaussian for each weight, which is directly generated by the TC. In SM C.3.2 in Ref. [40], we also report results for another instance of explicit distribution.

POSTERIOR-REPLAY WITH IMPLICIT DISTRIBUTIONS. Since the expressivity of *explicit* distributions is limited, we also explore the more diverse variational family of *implicit* distributions [146, 170]. These are parametrized by a WG that now takes the form of an auxiliary neural network, making the parameters $\boldsymbol{\theta}^{(t)}$ of the approximate posterior dependent on the chosen WG architecture. This setting, referred to as **PosteriorReplay-Imp**, results in a hierarchy of three networks: a TC network generates task-specific parameters $\boldsymbol{\theta}^{(t)}$ for the approximate posterior, which is defined through an arbitrary base distribution $p(\boldsymbol{z})$ and the WG hypernetwork, which in turn generates weights $\boldsymbol{w}$ for a main network M that processes the actual inputs of the dataset $\mathcal{D}^{(t)}$. Interestingly, the TC now plays the role of a *hyper-hypernetwork* as it generates the weights of another hypernetwork (Fig. 6.2a and Fig. 6.2c).

Variational inference commonly resorts to optimizing an objective consisting of a data-dependent term and a *prior-matching term* $\mathrm{KL}(q_{\boldsymbol{\theta}}(\boldsymbol{w}); p(\boldsymbol{w}))$ (cf. Eq. 3.16). Estimating the *prior-matching term* when using implicit distributions is not straightforward since we do not have analytic access to the density nor the entropy of $q_{\boldsymbol{\theta}^{(t)}}(\boldsymbol{w})$. To overcome this challenge, we resort to the spectral Stein gradient estimator (SSGE, see SM C.4.2 of Ref. [40] or the original work by Shi, Sun & Zhu [82]). This method is based on the insight that direct access to the log-density is not required, but only to its gradient with respect to $\boldsymbol{w}$. Noticing that this quantity appears in Stein's identity, the authors consider a spectral decomposition of the term and use the Nyström method to approximate the eigenfunctions. We test an alternative method for dealing with implicit distributions in SM C.4.1 of Ref. [40] that is based on estimating the log-density ratio.

As an additional challenge introduced by the use of implicit distributions, the support of $q_{\boldsymbol{\theta}^{(t)}}(\boldsymbol{w})$ is limited to a low-dimensional manifold when using an inflating architecture for WG, causing the *prior-matching term* to be ill-defined. To overcome this, we investigate the use of small noise perturbations in WG outputs (see SM C.4.3 of Ref. [40] for details). Normalizing flows [171] can also be utilized as WG architectures to gain analytic access to $q_{\boldsymbol{\theta}^{(t)}}(\boldsymbol{w})$, albeit at the cost of architectural constraints such as invertibility.

OVERCOMING FORGETTING VIA META-REPLAY. Since all learnable parameters are part of the TC system, forgetting only needs to be addressed at this meta-level. With $\mathcal{L}_{\text{task}}$ the task-specific loss (e. g., the ELBO, cf. Eq. 4.11) and $\mathrm{D}(\cdot;\cdot)$ a divergence measure between distributions, the loss for task t becomes:

$$\mathcal{L}^{(t)}(\boldsymbol{\psi}, \mathcal{E}, \mathcal{D}^{(t)}) = \mathcal{L}_{\text{task}}(\boldsymbol{\psi}, \boldsymbol{e}^{(t)}, \mathcal{D}^{(t)}) + \beta \sum_{t'<t} \mathrm{D}\big(q_{\boldsymbol{\theta}^{(t',*)}}(\boldsymbol{w}); q_{\boldsymbol{\theta}^{(t')}}(\boldsymbol{w})\big) \quad (6.1)$$

where $\mathcal{E} = \{\boldsymbol{e}^{(t')}\}_{t'=1}^{t}$ is the set of task embeddings up to the current task, β is the strength of the regularizer, $\boldsymbol{\theta}^{(t')} = f_{\text{TC}}(\boldsymbol{e}^{(t')}; \boldsymbol{\psi})$ depends on the parameters $\boldsymbol{\psi}$ and $\boldsymbol{e}^{(t')}$ to be learned, and $\boldsymbol{\theta}^{(t',*)} = f_{\text{TC}}(\boldsymbol{e}^{(t',*)}; \boldsymbol{\psi}^*)$ are the parameters of the posterior approximation obtained from a checkpoint of the TC parameters before learning task t: $\{\boldsymbol{\psi}^*\} \cup \{\boldsymbol{e}^{(t',*)}\}_{t'<t}$. The checkpointed meta-model allows replaying posterior distributions from the past and retaining them via divergence minimization as detailed below, hence the name *posterior meta-replay*. Importantly, the loss on task t only depends on the corresponding dataset $\mathcal{D}^{(t)}$, but knowledge transfer across tasks is

possible because task-specific models are learned through a shared meta-model.[1] Since the computation required to compute this regularizer linearly scales with the number of tasks, we also explore stochastically regularizing on a subset of randomly selected tasks in each update (cf. SM D.3 of Ref. [40]), and show that this does not impair performance. Notably, our *PosteriorReplay* method does not incur in a significant increase of runtime or memory usage (see SM F.2 of Ref. [40] for details).

The evaluation of Eq. 6.1 requires estimates of a divergence measure to prevent changes in learned posterior approximations of past tasks. Because these are required at every loss evaluation and need to be cheap to compute, we do not consider sample-based estimates but only estimates that directly utilize posterior parameters. This goal is easy to achieve for families of posterior approximations that possess an analytic expression for a divergence measure (e.g., Gaussian distributions). More specifically, for *PosteriorReplay-Exp*, we consider the forward KL, backward KL and the 2-Wasserstein distance but did not observe that the specific choice of divergence measure is crucial in practice (cf. SM C.3.1 and Table S14 of Ref. [40]). In all other cases, approximations are required. Specifically, we resort in our experiments to the use of an L2 regularizer at the output of the TC network:

$$\beta \sum_{t'<t} \| f_{\mathrm{TC}}(\boldsymbol{e}^{(t',*)}, \boldsymbol{\psi}^*) - f_{\mathrm{TC}}(\boldsymbol{e}^{(t')}, \boldsymbol{\psi}) \|_2^2 \tag{6.2}$$

Perhaps surprisingly, we observe that this crude regularization, reminiscent to the one used in the previous chapter (Eq. 5.3) for point estimates, does not harm performance and leads to models that exhibit virtually no forgetting. However, we discuss in SM F3 of Ref. [40] how this isotropic regularization could be improved given that the KL is locally approximated by a norm $\|\cdot\|_F$ induced by the Fisher information matrix F on $q_{\boldsymbol{\theta}}(\boldsymbol{w})$.

TASK INFERENCE. A system with task-specific solutions requires access to task identity when processing unseen samples (cf. Sec. 5.2.3). In our framework, this amounts to selecting the correct task embedding to condition the TC. Although auxiliary systems can be used to infer task identity (e.g., see Ref. [38] or HNET+TIR from the previous chapter), here we exploit predictive uncertainty, assuming task identity can be inferred from the input alone. For a task inference approach based on predictive

1 While this type of transfer is rather implicit, explicit knowledge transfer can be realized via task-specific priors (cf. SM F.7 of Ref. [40]).

uncertainty to work, the properties of the input data distribution $p^{(t)}(\boldsymbol{x})$ need to be reflected in the uncertainty measure, i.e., uncertainty needs to be low for in-distribution data and high for out-of-distribution (OOD) data (we discuss this topic in detail in Chapter 7). Uncertainty-based task inference with deterministic discriminative models only captures aleatoric uncertainty, making its overall validity debatable. Indeed, aleatoric uncertainty is only calibrated in-distribution and its OOD behavior is hard to foresee (cf. Sec. 3.1). Instead, we argue that epistemic uncertainty arising naturally in a Bayesian setting is crucial for robust uncertainty-based task inference. In our case, epistemic uncertainty stems from the fact that the posterior parameter distribution $p(\boldsymbol{w} \mid \mathcal{D}^{(t)})$ in conjunction with the network architecture induces a distribution over functions. If this distribution captures a rich set of functions, a diversity of predictions on OOD data can be expected even if those functions agree in-distribution (cf. Sec. 6.4.2). Note, however, that inducing such diverse distribution over functions is not straightforward with neural networks, and that more research is required to justify uncertainty-based OOD detection as we will detail in Chapter 7.

We explore two different ways to quantify uncertainty for task inference. In **Ent** the task t leading to the lowest entropy on the predictive posterior $p(\boldsymbol{y} \mid \mathcal{D}^{(t)}, \tilde{\boldsymbol{x}})$ is selected, where $p(\boldsymbol{y} \mid \mathcal{D}^{(t)}, \tilde{\boldsymbol{x}}) = \int_{\boldsymbol{w}} p(\boldsymbol{y} \mid \boldsymbol{w}, \tilde{\boldsymbol{x}}) p(\boldsymbol{w} \mid \mathcal{D}^{(t)})\, d\boldsymbol{w}$ is approximated via Monte-Carlo with samples from $q_{\boldsymbol{\theta}^{(t)}}(\boldsymbol{w})$. This approach captures both aleatoric and epistemic uncertainty when used in a probabilistic setting. In **Agree** the task leading to the highest agreement in predictions across models drawn from $q_{\boldsymbol{\theta}^{(t)}}(\boldsymbol{w})$ is selected. This approach exclusively measures epistemic uncertainty and can therefore only be estimated in a probabilistic setting (cf. Sec. 3.4). Although we generally consider task inference for individual samples, we also explore batch-wise (**BW**) task inference for batches of 100 samples that are assumed to belong to the same task. Such approach drastically boosts task inference simply due to a statistical accumulation effect when having above chance level task inference for single inputs. Intuitively, *BW* corresponds to accumulating evidence to decrease uncertainty (e.g., an agent looking at an object from multiple perspectives). Further details can be found in SM C.6 of Ref. [40], and using uncertainty for task-boundary detection when training without explicit access to task identity is explored in SM D.8 of Ref. [40].

FACILITATING TASK INFERENCE THROUGH CORESETS. A key advantage of Bayesian statistics is the ability to update models as new evidence arrives. When continually learning a sequence of tasks, posteriors may

for example undergo a post-hoc fine-tuning on stored coresets to mitigate catastrophic forgetting of earlier tasks. Specifically, given a dataset split $\mathcal{D}^{(t)} \setminus \mathcal{C}^{(t)} \cup \mathcal{C}^{(t)}$, one can perform a final update $p(\boldsymbol{w} \mid \mathcal{D}^{(t)}) \propto p(\boldsymbol{w} \mid \mathcal{D}^{(t)} \setminus \mathcal{C}^{(t)}) p(\mathcal{C}^{(t)} \mid \boldsymbol{w})$ using a stored coreset $\mathcal{C}^{(t)}$ in conjunction with an already learned posterior approximation for $p(\boldsymbol{w} \mid \mathcal{D}^{(t)} \setminus \mathcal{C}^{(t)})$ that now acts as a prior. Interestingly, access to coresets after training on all tasks can also be exploited to facilitate task inference via predictive uncertainty. Here, we explore this idea by encouraging task-specific models to produce uncertain predictions for OOD samples (i.e., coresets from other tasks), an approach that we denote **CS** (see SM C.7 of Ref. [40] for details), and in which we store 100 inputs per task. Intriguingly, training on OOD inputs makes these become in-distribution, and therefore renders model agreement (*Agree*) inapplicable for task inference, which we empirically observe.

IMPROVING PRIOR-FOCUSED CL. We investigate ways to improve *Prior-Focused* methods, as introduced in Sec. 4.1, within our framework. First, we endow them with *implicit* posterior approximations $q_{\boldsymbol{\theta}}(\boldsymbol{w})$ parametrized by a WG hypernetwork, an approach we refer to as **PriorFocused-Imp**. Because the posterior is shared across tasks, no TC system is required and the parameters $\boldsymbol{\theta}$ can be directly optimized via SSGE. Second, we enrich *PriorFocused* methods with a small set of task-specific parameters that enable uncertainty-based task inference for prior-focused methods too. Specifically, the learned parameters $\boldsymbol{w}$ consist of a set of shared weights ϕ and a set of task-specific output heads with weights $\{\boldsymbol{\xi}^{(t)}\}_{t=1}^{T}$. This approach is in contrast with how *PriorFocused* methods like *Online EWC* are commonly deployed in task-agnostic inference settings, where the softmax output grows as new tasks arrive (e.g., [106]). The use of a growing softmax causes the model class parametrized by $\boldsymbol{w}$ to change over time, and therefore violates the Bayesian assumption that the approximate posterior is obtained from a model class containing the ground-truth model. We show that this leads to limitations that can be overcome by a multihead approach. For *Online EWC*, we refer to the growing softmax and multihead scenarios as **EWC-growing** and **EWC-multihead**, respectively (cf. SM C.5.2 of Ref. [40]). We also explore the prior-focused instantiation of BbB, known as **VCL** (cf. SM C.5.1 of Ref. [40]).

6.4 EXPERIMENTS

In this section, we start by illustrating the conceptual advantage of the *PosteriorReplay* approach compared to *PriorFocused* methods, as well as the importance of parameter uncertainty for robust task inference. We then explore scalability to more challenging computer vision CL benchmarks.

To assess forgetting, we provide **During** scores, measured directly after training each task, and **Final** scores, evaluated after training on all tasks. We consider two different testing scenarios: (1) either task-identity is explicitly given (**TGiven**) or (2) task-identity has to be inferred (**TInfer**), e. g., via predictive uncertainty. Unless explicitly mentioned, task inference is performed for each sample point in isolation and is obtained using the *Ent* criterion (*TInfer-Final*). Note, that one may interpret *TGiven-Final* as `CL1` and *TInfer-Final* as `CL3`, respectively (cf. 4.1). Whenever the wrong task is inferred, the sample point is directly considered as incorrectly classified. Supplementary results and controls are provided in SM D of Ref. [40], and all experimental details can be found in SM E of Ref. [40].[2]

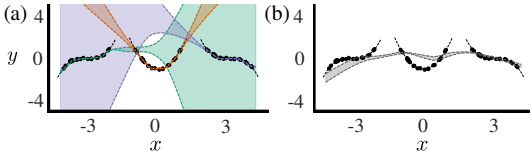


FIGURE 6.3: *PriorFocused* methods struggle to learn three 1D polynomial regression tasks. **(a)** Different colors represent the task-specific posterior approximations within the final *PosteriorReplay-Exp* model. For unseen inputs x the posterior with the lowest predictive uncertainty is chosen to make predictions. **(b)** Predictive posterior using the final approximation of $p(\boldsymbol{w} \mid \mathcal{D}^{(1:3)})$, obtained via *PriorFocused-Imp*. Shaded areas represent standard deviation, and black dots training samples.

6.4.1 *Simple 1D regression illustrates the pitfalls of prior-focused learning*

To illustrate conceptual differences between *PosteriorReplay* and *PriorFocused* methods we study 1D regression, for which the predictive posterior can be visualized. Each task-specific posterior obtained with *PosteriorReplay-Exp* fits the training data well (Fig. 6.3a) and exhibits increasing uncertainty when leaving the in-distribution domain, as desired for successful task

2 Source code for all experiments (including all baselines) is available at: `https://github.com/chrhenning/posterior_replay_cl`.

Final Acc	TGiven	TInfer (Ent)	TInfer (Agree)
PR-Dirac	99.78 ± 0.21	44.90 ± 5.74	N/A
PR-Exp	100.0 ± 0.00	81.07 ± 6.78	90.02 ± 3.57
PR-Imp	100.0 ± 0.00	100.0 ± 0.00	100.0 ± 0.00

TABLE 6.1: 2D mode classification accuracies (Mean $\pm$ standard error of the mean (SEM) in %, $n = 10$). Task identity is inferred via predictive uncertainty using an entropy (*Ent*) or model agreement (*Agree*) criterion. *PR* denotes *PosteriorReplay*.

inference. Interestingly, when studying low-dimensional problems, we generally found it easier to find viable hyperparameter configurations for *PosteriorReplay* with *implicit* methods than with *explicit* ones. As we do not consider a multihead for this problem, *PriorFocused* methods have to fit a single posterior to all polynomials in a sequential manner and struggle to find a good fit (Fig. 6.3b), independent of the type of posterior approximation used. Results for other methods and an analysis of the correlations and multi-modality that can be captured by implicit methods in weight space can be found in SM D.1 of Ref. [40].

Because we use a mean squared error (MSE) loss, the likelihood is a Gaussian with fixed variance (cf. Example 2) and all $\boldsymbol{x}$-dependent uncertainty originates from parameter uncertainty. We next consider classification problems where both epistemic and aleatoric uncertainty can be explicitly modelled.

6.4.2 *Maintaining parameter uncertainty is crucial for robust task inference*

To investigate the importance of parameter uncertainty, we consider a 2D classification problem for which uncertainty can be visualized in- and out-of-distribution. Classification tasks are of special interest as it is possible to model arbitrary input-dependent discrete distributions, e.g. via a softmax at the outputs. This surprisingly often results in meaningful OOD performance in high-dimensional benchmarks without any treatment of parameter uncertainty [49].

Here, we consider a Gaussian mixture of two modes per task, each mode being a different class (Fig. 6.4a). *TInfer-Final (Agree)*, which is indicative of the importance of epistemic uncertainty for OOD detection, is the most robust measure for task inference in this experiment (Table 6.1).

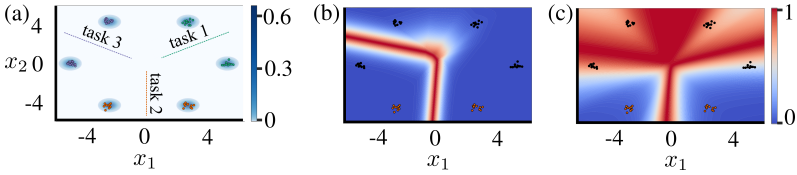


FIGURE 6.4: Parameter uncertainty is crucial for robust task inference. **(a)** Density of input distribution $p(\boldsymbol{x})$ across tasks. Dots represent training points, colors task-affiliation and lines decision boundaries for each of the three consecutively learned 2D binary classification tasks. **(b)** Entropy of predictive distribution induced by the approximate posterior of task 2 learned via *PosteriorReplay-Dirac*. **(c)** Same as (b) for *PosteriorReplay-Imp*.

PosteriorReplay-Dirac, which does not incorporate epistemic uncertainty, performs poorly. Finally, *implicit* methods maintain an advantage over *explicit* ones, presumably due to the increased flexibility in modelling the posterior, but see Chapter 7 for the importance of the prior for OOD detection.

To better understand why differences between methods arise we provide uncertainty maps over the input space (Fig. 6.4). Consistent with the observed low task inference accuracy, *PosteriorReplay-Dirac* displays arbitrary uncertainty away from the training data (Fig. 6.4b). By contrast, *PosteriorReplay-Imp* (Fig. 6.4c) approaches the desired behavior of displaying high uncertainty away from the training data of the corresponding task. We provide detailed analysis in SM D.2 of Ref. [40].

6.4.3 *Multiple factors affect uncertainty-based task inference accuracy*

To investigate the factors that affect uncertainty-based task inference, we next consider SplitMNIST [103], which is an MNIST adaptation for CL by splitting the ten digit classes into five binary classification tasks (cf. Fig. 4.1). The results can be found in Table 6.2.

While all methods successfully prevent forgetting (i. e., *Final* scores are maxed-out and close to the *During* accuracies, see SM D.3 in Ref. [40]) and achieve acceptable *Final* accuracies when task identity is provided, large differences can be observed when the task needs to be inferred. Methods with task-specific solutions outperform by a large margin *Prior-Focused* approaches such as *Online EWC*, whose performance substantially improves when using uncertainty-based task inference through a multi-head. Despite superior performance of *PosteriorReplay* approaches, a gap

	TGiven-Final	TInfer-Final
EWC-growing [105]	N/A	19.96 ± 0.07
EWC-multihead	96.40 ± 0.62	47.67 ± 1.52
VCL-multihead	96.45 ± 0.13	58.84 ± 0.64
PR-Dirac	99.65 ± 0.01	70.88 ± 0.61
PR-Exp	99.72 ± 0.02	71.73 ± 0.87
PR-Imp	99.77 ± 0.01	71.91 ± 0.79
SP-Dirac	99.77 ± 0.01	70.39 ± 0.27
SP-Exp	99.81 ± 0.00	68.40 ± 0.23
PR-Exp-BW	99.72 ± 0.02	99.72 ± 0.02
PR-Exp-CS	98.50 ± 0.09	90.83 ± 0.24
DGR [105]	N/A	91.79 ± 0.32
HNET+R [39]	N/A	95.30 ± 0.13
PR-Dirac[1]	99.72 ± 0.01	63.41 ± 1.54
PR-Exp[1]	99.75 ± 0.01	70.07 ± 0.56
PR-Dirac[2]	99.87 ± 0.04	72.33 ± 2.75
PR-Exp[2]	99.20 ± 0.67	74.09 ± 1.38

TABLE 6.2: Accuracies of SplitMNIST experiments (Mean ± SEM in %, $n = 10$) after learning all tasks when task identity is provided (*TGiven-Final*) and when it needs to be inferred (*TInfer-Final*, based on the *Ent* criterion if explicit task-inference is required). Results are shown for an MLP with two hidden layers of 400 neurons (MLP-400,400), an MLP-100,100[1] or a Lenet[2]. *PR* denotes *PosteriorReplay* and *SP SeparatePosteriors*.

in performance between task-inferred and task-given scenarios remains. However, training separate posteriors that are not embedded in a hypernetwork (**SeparatePosteriors**) leads to similar results, showing that task inference limitations are not linked to our solution. These limitations can be surmounted by inferring the task on batches rather than single samples (BW) or by using coresets to encourage high uncertainty for OOD data (CS), which leads to performances comparable to generative-replay methods which explicitly capture $p(\mathbf{X})$ (i. e., HNET+R and DGR).

To better understand the factors that influence task inference, we consider a variety of approximate inference methods and architectures. Since epistemic uncertainty seems to play a vital role for task inference, the approximate inference method will likely affect *TInfer* performance (e. g., *Exp* vs. *Imp*; in supplementary results). In addition, because diversity in function space enables uncertainty-based OOD detection and because different architectures induce different priors in function space [67], one can expect that prior and architecture play a key role as well, which we observe by comparing the *TInfer* performance for different architectures. Additional SplitMNIST results can be found in SM D.3 of Ref. [40], and results showing scalability to sequences of up to 100 PermutedMNIST tasks can be found in SM D.4 and D.5 of Ref. [40].

6.4.4 *PosteriorReplay scales to natural image datasets*

While BNNs are advocated because of their theoretical promises, practitioners are often put off by scalability issues. Here we show that our approach scales to natural images by considering SplitCIFAR-10 [172], a dataset consisting of five tasks with two classes each. Results obtained with a Resnet-32 [147] (Table 6.3) show performance gains in the task-agnostic setting compared to recent methods like EBM [108], and to the *PriorFocused* method VCL.

Furthermore, our results reveal considerable improvements through the incorporation of epistemic uncertainty, as shown by differences between *PosteriorReplay-Exp* and *PosteriorReplay-Dirac*.

Notably, *PosteriorReplay-Exp-BW* solves CIFAR-10 with a performance comparable to a classifier trained on all data at once, with the caveat that successive unseen samples are assumed to belong to the same task. In contrast to low-dimensional problems, the *implicit* method *PosteriorReplay-Imp* does not exhibit a competitive advantage, as it appears to suffer from scalability issues. Other baselines and results for a WRN-28-10 can be found

	TGiven-During	TGiven-Final	TInfer-Final
VCL-multihead	95.78 ± 0.09	61.09 ± 0.54	15.97 ± 1.91
PR-Dirac	94.59 ± 0.10	93.77 ± 0.31	54.83 ± 0.79
PR-Exp	95.59 ± 0.08	95.43 ± 0.11	61.90 ± 0.66
PR-Imp	94.25 ± 0.07	92.83 ± 0.16	51.95 ± 0.53
PR-Exp-BW	95.59 ± 0.08	95.43 ± 0.11	92.94 ± 1.04
PR-Exp-CS	95.15 ± 0.11	92.48 ± 0.13	64.76 ± 0.34
EBM [108]	N/A	N/A	38.84 ± 1.08

TABLE 6.3: Accuracies of SplitCIFAR-10 experiments (Mean $\pm$ SEM in %, $n = 10$). *TInfer-Final* is based on the *Ent* criterion if explicit task-inference is required and *PR* denotes *PosteriorReplay*.

in SM D.6 of Ref. [40], and results showing that our framework scales to the SplitCIFAR-100 benchmark can be found in SM D.7 of Ref. [40].

6.5 DISCUSSION

In this chapter we propose *posterior meta-replay*, a framework for continually learning task-specific posterior approximations within a single shared meta-model. In contrast to *prior-focused* methods based on a recursive Bayesian update, our approach does not directly seek trade-off solutions across tasks. This results in more flexibility for learning new tasks but introduces the need to know task identity when processing unseen inputs.

Task Inference. Probabilistic inference on task identity can be achieved by additionally considering inputs and task embeddings as random variables, a strategy that would require task-conditioned generative models with tractable density [125] (cf. Sec. 4.1.4). However, learning generative models on high-dimensional data is a challenging problem and, even if tractable densities are accessible, these do not currently reflect the underlying data-generative process [126].[3] To sidestep these limitations, we study the use of predictive uncertainty for task inference [39] and show that an entropy-based criterion works best for both deterministic and Bayesian models.

3 Interestingly, a concurrent study by van de Ven, Li & Tolias [124] successfully trains class-conditioned generative models, indicating that this approach could nevertheless be feasible to tackle task inference.

Nevertheless, we highlight that proper task inference requires epistemic uncertainty (e. g., measured in terms of model disagreement, cf. Sec. 3.4). Indeed, in-distribution sample points with high aleatoric uncertainty can lead to high predictive entropy, causing them to be misclassified as OOD. This does not pose a problem in highly-curated ML datasets where samples with high aleatoric uncertainty are excluded [173], but drastically limits the applicability of entropy-based uncertainty estimation in more practical scenarios. For these reasons, we advocate for the use of Bayesian models whose epistemic uncertainty can induce diversity in function space for OOD inputs and enable more robust task inference.[4]

Limitations. Compared to methods performing deterministic inference, the Bayesian model average incurs in significant computational overhead. This overhead is reinforced when performing uncertainty-based task inference, since each predictive posterior needs to be evaluated in parallel. Moreover, despite strong performance gains compared to *prior-focused* approaches, we observe general limitations of such task inference procedure. These could be overcome once a better understanding of how epistemic uncertainty influences OOD behavior in neural networks is available (cf. Chapter 7). In addition, our work builds on algorithms to perform variational inference, and is therefore only applicable to problems where these can be successfully deployed. Finally, all our experiments consist of a set of clearly defined tasks within which i.i.d. samples are available. Although this scenario is in line with most existing CL literature, it might be of limited relevance for practical CL problems, and a focus on established benchmarks adhering to these constraints could therefore misguide research on this area. Indeed, a more natural CL problem might arise from the need to online learn from a stream of autocorrelated samples (cf. Sec. 4.2). In this context, it is important to note that unlike non-Bayesian CL methods, our approach can utilize any type of online *prior-focused* method (such as FOO-VB [131]) to also learn *within* tasks in a non-i.i.d. manner. Therefore, as long as some coarse split into tasks is meaningful, such hierarchical approach holds great promise. However, it should be noted that any progress towards learning from non-i.i.d. data opens the door to training algorithms from raw, uncurated datasets, and could therefore counter some of the efforts that are currently done to mitigate algorithmic bias.

4 Note, while also models with deterministic parameters may be well suited for OOD detection (e.g., see work by Lakshminarayanan *et al.* [174], that utilizes a deterministic distance preserving input-to-hidden mapping), these solutions are limited by the fact that the Bayesian recursive update is not applicable and therefore parameters cannot be updated in a sound way when learning continually.

Conclusion. Taken together, our work shows that it is possible to continually learn an approximate posterior per task without an increased parameter budget, and that task-agnostic inference can be achieved via predictive uncertainty to obtain a Bayesian CL approach that is scalable to real-world data.

In particular, we could improve upon the HNET+ENT baseline from Chapter 5 (the only proposed baseline without the need for generative modelling), and utilize the added Bayesian perspective to propose extensions such as using coresets for explicitly improving task inference.

Since forgetting is not a prevalent issue in our experiments and task inference limitations are not linked to our CL solution, progress in the field of uncertainty-based OOD detection will automatically translate into further improvements of our method. For this reason, we use the next chapter to gain further insights into the problem of uncertainty-based OOD detection.

7

UNCERTAINTY-BASED OUT-OF-DISTRIBUTION DETECTION REQUIRES SUITABLE FUNCTION SPACE PRIORS

This chapter's content is taken from an online preprint ***authored by****: Francesco D'Angelo* and Christian Henning* [95].*

** These authors contributed equally.*

Better uncertainty-based *out-of-distribution* (OOD) detection can improve task inference as studied in the previous chapters. But the OOD problem is more general and as discussed in Chapter 3, the need to avoid confident predictions on unfamiliar data sparks interest in this topic. It is widely assumed that Bayesian neural networks (BNNs) are well suited for this task, as the endowed epistemic uncertainty should lead to disagreement in predictions on outliers. In this chapter, we question this assumption and show that proper Bayesian inference with function space priors induced by neural networks does not necessarily lead to good OOD detection. To circumvent the use of approximate inference, we start by studying the infinite-width case, where Bayesian inference can be exact due to the correspondence with Gaussian processes. Strikingly, the kernels induced under common architectural choices lead to distributions over functions which cause predictive uncertainties that do not reflect the underlying data generating process and are therefore unsuited for OOD detection. Importantly, we find this OOD behavior to be consistent with the corresponding finite-width networks. To overcome this limitation, useful function space properties can also be encoded in the prior in weight space, however, this can currently only be applied to a specified subset of the domain and thus does not inherently extend to OOD data. Finally, we argue that a trade-off between generalization and OOD capabilities might render the application of BNNs for OOD detection undesirable in practice. Overall, our study discloses fundamental problems when naively using BNNs for OOD detection and opens interesting avenues for future research.

7.1 INTRODUCTION

One of the challenges that the modern machine learning community is striving to tackle is the detection of unseen inputs for which predictions should not be trusted. This problem is also known as out-of-distribution (OOD) detection. The challenging nature of this task is partly rooted in the fact that there is no universal mathematical definition that characterizes an unseen input $\boldsymbol{x}^*$ as OOD as discussed in Sec. 7.2. Without such a definition, there is no foundation for deriving detection guarantees under verifiable assumptions. In this work, we consider OOD points as points that are unlikely under the distribution of the data $p(\boldsymbol{x})$ and points that are outside the support of $p(\boldsymbol{x})$. Importantly, we do not claim to provide a meaningful definition of OOD, but rather argue that the justification of an OOD method can only be assessed theoretically once such a definition is provided. While it appears natural to tackle the OOD detection problem from a generative perspective by explicitly modelling $p(\boldsymbol{x})$, this paper is solely concerned with the question of how justified it is to deploy the predictive uncertainty of a Bayesian neural network (BNN) for OOD detection.

As recent developments forecast an increasing integration of deep learning methods into industrial applications, it becomes essential to provide the theoretical groundings that justify the use of uncertainty for OOD detection, a task that is crucial for safety-critical applications of AI and reliable prediction-making.

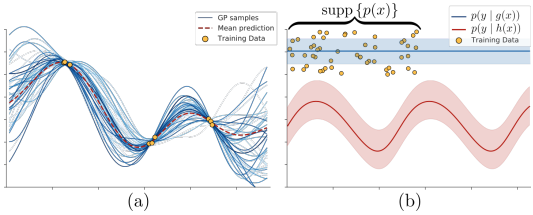


FIGURE 7.1: **(a)** GP regression with an RBF kernel illustrates uncertainty-based OOD detection. The prior variance over function values is only squeezed around training points, which leaves epistemic uncertainty high in OOD regions. **(b)** Conceptual illustration on why epistemic uncertainty is not necessarily linked to OOD detection (see main text for more details). Note, that the ground-truth $p(\boldsymbol{y} \mid \boldsymbol{x})$ is only defined within the support of $p(\boldsymbol{x})$.

When BNNs are used in supervised learning, the dataset is composed of inputs $\boldsymbol{x} \in \mathcal{X}$ and targets $\boldsymbol{y} \in \mathcal{Y}$ which are assumed to be generated according to some unknown process: $\mathcal{D} \overset{i.i.d.}{\sim} p(\boldsymbol{x})p(\boldsymbol{y} \mid \boldsymbol{x})$ (cf. Def. 2). The goal of learning is to infer from $\mathcal{D}$ alone the distribution $p(\boldsymbol{y} \mid \boldsymbol{x})$ in order to make predictions on unseen inputs $\boldsymbol{x}^*$ (cf. Def. 3). In the case of deep learning, this

problem is approached by choosing a neural network $f(\cdot; \boldsymbol{w})$ parametrized by $\boldsymbol{w}$, and predictions are made via the conditional $p\big(\boldsymbol{y} \mid f(\boldsymbol{x}; \boldsymbol{w})\big)$ (also called the *likelihood* of $\boldsymbol{w}$ for given $\boldsymbol{x}, \boldsymbol{y}$). Assuming that the induced class of hypotheses contains some $\hat{\boldsymbol{w}}$ such that $p\big(\boldsymbol{y} \mid f(\boldsymbol{x}; \hat{\boldsymbol{w}})\big) = p(\boldsymbol{y} \mid \boldsymbol{x})$ almost everywhere on the support of $p(\boldsymbol{x})$, Bayesian statistics can be used to infer plausible models $p(\boldsymbol{w} \mid \mathcal{D})$ under the observed data given some prior knowledge $p(\boldsymbol{w})$ (see Sec. 3.2 for an introduction). This parametric description together with the network implicitly induces a prior over functions $p(\boldsymbol{f})$ [65, 175]. Believing in the validity of a subjective choice of prior [176], Bayesian inference comes with a multitude of benefits as it is less susceptible to overfitting, allows to incorporate new evidence without requiring access to past data (cf. Sec. 4.5) and provides interpretable uncertainties (but see Sec. 3.3).

The uncertainty captured by a BNN can be coarsely categorized into *aleatoric* and *epistemic* uncertainty (cf. Sec. 3.1). Aleatoric uncertainty is irreducible and intrinsic to the data $p(\boldsymbol{y} \mid \boldsymbol{x})$. On the other hand, a BNN also models epistemic uncertainty by maintaining a distribution over parameters $p(\boldsymbol{w} \mid \mathcal{D})$.[1] This distribution reflects the uncertainty about which hypothesis explains the data and can be reduced by observing more data. Arguably, aleatoric uncertainty is of little interest for detecting OOD inputs. However, the recent advent of overparametrized models which further extend the hypothesis class, deceptively motivated the idea that the epistemic uncertainty under a Bayesian framework might be intrinsically suitable to detect unfamiliar inputs, and therewith implying, that BNNs can be used for OOD detection. This conjecture is intuitively true if the following assumption holds: the hypotheses captured by $p(\boldsymbol{w} \mid \mathcal{D})$ must agree on their predictions for in-distribution samples but disagree for OOD samples (Fig. 7.1a). Certain Bayesian methods satisfy this assumption, e.g. a Gaussian process regression with an RBF kernel (cf. Sec. 7.4). However, the uncertainty induced by Bayesian inference does not in general give rise to OOD capabilities. This can easily be verified by considering the following thought experiment (Fig. 7.1b): Assume a model class with only two hypotheses $g(x)$ and $h(x)$ such that $g(x) \neq h(x) \, \forall x$ and data being generated according to $g(x)$ on a restricted domain. Once data is observed, we can commit to the ground-truth hypothesis $g(x)$ and thus lose epistemic uncertainty in- and out-of-distribution alike. Finite-width neural networks, by contrast, form a powerful class of models, and are often put into context

1 Note, that the Bayesian treatment of network parameters does not account for all types of epistemic uncertainty such as the uncertainty stemming from model misspecification (cf. Fig. 3.2, [46]). We, however, assume the model to be correctly specified.

with universal function approximators [177]. But is this fact in combination with Bayesian statistics enough to attribute them with good OOD capabilities? The literature often seems to imply that a BNN is intrinsically good at OOD detection. For instance, the use of OOD benchmarks when introducing new methods for approximate inference creates the false impression that the true posterior is a good OOD detector [9, 41–44, 178, 179]. Our work is meant to start a discussion among researchers about the properties of OOD uncertainty of the exact Bayesian posterior of neural networks. To initiate this, we contribute as follows:

- We emphasize the importance of the prior in function space for OOD detection, which is induced by the choice of architecture (Sec. 7.4) and weight space prior (Sec. 7.5).
- We empirically show that exact inference in infinite-width networks under common architectural choices does not necessarily lead to desirable OOD behavior, and that these observations are consistent with results obtained via approximate inference in their finite-width counterparts.
- We furthermore study the OOD behavior in infinite-width networks by analysing the properties of the induced kernels. Moreover, we discuss desirable kernel features for OOD detection and show that also neural networks can approximately carry these features.
- We emphasize that the choice of weight-space prior has a strong effect on OOD performance, and that encoding desirable function space properties within unknown OOD domains into such prior is challenging.
- We argue that there is a trade-off between good generalization and having high uncertainty on OOD data. Indeed, improving generalization by incorporating prior knowledge (which is usually encoded in an input-domain agnostic manner) can negatively impact OOD uncertainties.

7.2 ON THE DIFFICULTY OF DEFINING OUT-OF-DISTRIBUTION INPUTS

While the intuitive notion of OOD sample points (or outliers) is commonly agreed upon, for instance as "an observation (or subset of observations) which appears to be inconsistent with the remainder of that set of data" [180], a mathematical formalization of the essence of an outlier is difficult (cf. SM A.1) and requires subjective characterizations [181]. Often, methods for outlier detection are designed based on an intuitive notion (see the work

by Pimentel *et al.* [182] for a review), and therewith only implicitly define a method-specific definition of OOD points. This also accounts for uncertainty-based OOD detection with neural networks, where a statistic of the predictive distribution that quantifies uncertainty (e.g., the entropy) is used to decide whether an input is considered OOD [144]. Commonly, the parameters $\boldsymbol{w}$ of a neural network are trained using an objective derived from $\mathbb{E}_{p(\boldsymbol{x})}\left[\mathrm{KL}\left(p(\boldsymbol{y} \mid \boldsymbol{x}); p\big(\boldsymbol{y} \mid f(\boldsymbol{x}; \boldsymbol{w})\big)\right)\right]$ (cf. Examples 2 and 3). Therefore, these networks only have calibrated uncertainties in-distribution with OOD uncertainties not being controlled for unless explicit training on OOD data is performed [eg., 183]. The question this study is concerned with is therefore: *does the explicit treatment of parameter uncertainty via the posterior parameter distribution $p(\boldsymbol{w} \mid \mathcal{D})$ elicit provably high uncertainty OOD?*

Note, that we do not consider the problem of outlier detection within the training set [181], but rather ask whether a deployed model is able to *know what it does not know*.

7.3 BACKGROUND

In this section, we briefly introduce the concepts on which we base our argumentation in the coming sections. We start by recapping BNNs, which rely on approximate inference. We then discuss that in the non-parametric limit and under a certain choice of prior, a BNN converges to a Gaussian process, an alternative Bayesian inference framework where exact inference is possible. The connection between BNNs and Gaussian processes will later allow us to make interesting conjectures about OOD behavior. Finally, we introduce generalization bounds from the PAC-Bayes framework, which we will later use to argue that generalization and OOD detection can be conflicting objectives.

7.3.1 *Bayesian neural networks*

An introduction into this topic is provided in Chapter 3. We briefly recap the notation in this section.

In supervised deep learning, we typically construct a likelihood function from the conditional density $p\left(\boldsymbol{y} \mid f(\boldsymbol{x}; \boldsymbol{w})\right)$, parameterized by a neural network $f(\boldsymbol{x}; \boldsymbol{w})$, and the training data $\mathcal{D} = \{(\boldsymbol{x}^{(n)}, \boldsymbol{y}^{(n)})\}_{n=1}^{N}$. In BNNs, this is used to form the posterior distribution of all likely network parameterizations: $p(\boldsymbol{w} \mid \mathcal{D}) \propto \prod_{n=1}^{N} p\big(\boldsymbol{y}^{(n)} \mid f(\boldsymbol{x}^{(n)}; \boldsymbol{w})\big)\, p(\boldsymbol{w})$, where $p(\boldsymbol{w})$ is the prior distribution over weights. Crucially, when making a predic-

tion with the Bayesian approach on a test point $\boldsymbol{x}^*$, we do not only use a single parameter configuration $\hat{\boldsymbol{w}}$ but we marginalize over the whole posterior, thus taking all possible explanations of the data into account: $p(\boldsymbol{y}^* \mid \mathcal{D}, \boldsymbol{x}^*) = \int p(\boldsymbol{y}^* \mid f(\boldsymbol{x}^*; \boldsymbol{w}))\, p(\boldsymbol{w} \mid \mathcal{D})\, \mathrm{d}\boldsymbol{w}$.

7.3.2 *Gaussian process*

Gaussian processes are established Bayesian machine learning models that, despite their strong scalability limitations, can offer a powerful inference framework. Formally [61]:

Definition 7 *A Gaussian process (GP) is a collection of random variables, any finite number of which have a joint Gaussian distribution.*

Compared to parametric models, GPs have the advantage of performing inference directly in function space. A GP is defined by its mean $m(\boldsymbol{x}) = \mathbb{E}[f(\boldsymbol{x})]$ and covariance $C\big(f(\boldsymbol{x}), f(\boldsymbol{x}')\big) = \mathbb{E}\big[\big(f(\boldsymbol{x}) - m(\boldsymbol{x})\big)\big(f(\boldsymbol{x}') - m(\boldsymbol{x}')\big)\big]$. The latter can be specified using a kernel function $k : \mathbb{R}^{n_{\text{in}}} \times \mathbb{R}^{n_{\text{in}}} \to \mathbb{R}$ and implies a prior distribution over functions, which due to the marginalization properties of multivariate Gaussians can be consistently evaluated at any set of given input locations, e.g.:

$$p(\boldsymbol{f} \mid X) = \mathcal{N}\left(\boldsymbol{0}, K(X, X)\right) \tag{7.1}$$

where $K(X, X)_{ij} = k(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)})$ is the kernel Gram matrix on the training inputs X, $\boldsymbol{f}$ is the vector of function values at training locations and $m(\boldsymbol{x})$ has been chosen to be 0. When observing the training data, the prior is reshaped to place more mass in the regions of functions that are more likely to have generated them, and this knowledge is then used to make predictions on unseen inputs X_*. In probabilistic terms, this operation corresponds to conditioning the joint Gaussian prior: $p(\boldsymbol{f}_* \mid X_*, X, \boldsymbol{f})$. As commonly exercised in neural network regression, we assume to not have direct access to the function values but to noisy observations: $\boldsymbol{y} = \boldsymbol{f} + \epsilon$ with $\epsilon \sim \mathcal{N}(0, \sigma^2 \mathbb{I}_{n_{\text{in}}})$. This assumption is formally equivalent to a Gaussian likelihood $p(\boldsymbol{y} \mid \boldsymbol{f}) = \mathcal{N}(\boldsymbol{y} \mid \boldsymbol{f}, \sigma^2 \mathbb{I}_{n_{\text{in}}})$ (but see Chapter 3 and our work in Ref. [1] for a discussion on the pitfalls of this assumption). The conditional distribution on the noisy observation can then be written as [61]:

$$
\begin{aligned}
p(\boldsymbol{f}_* \mid X_*, X, \boldsymbol{y}) &= \int d\boldsymbol{f}\, p(\boldsymbol{f}_* \mid X_*, X, \boldsymbol{f}) p(\boldsymbol{f} \mid X, \boldsymbol{y}) = \mathcal{N}\left(\bar{\boldsymbol{f}}_*, C(\boldsymbol{f}_*)\right) \\
\bar{\boldsymbol{f}}_* &= K(X_*, X)[K(X, X) + \sigma^2 \mathbb{I}]^{-1} \boldsymbol{y} \\
C(\boldsymbol{f}_*) &= K(X_*, X_*) - K(X_*, X)[K(X, X) + \sigma^2 \mathbb{I}]^{-1} K(X, X_*)
\end{aligned}
\tag{7.2}
$$

where $p(\boldsymbol{f} \mid X, \boldsymbol{y}) = \frac{p(\boldsymbol{y}|\boldsymbol{f}) p(\boldsymbol{f}|X))}{p(\mathcal{D})}$ with $p(\mathcal{D})$ being the marginal likelihood.

RBF KERNEL. A commonly used kernel function for GPs is the squared exponential (RBF):

$$
k(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\frac{1}{2l^2} \|\boldsymbol{x} - \boldsymbol{x}'\|_2^2 \right) \quad , \tag{7.3}
$$

with the length scale l being a hyperparameter. It is important to notice that the covariance between outputs is written exclusively as a function of the distance between inputs. As a consequence, points that are close in the Euclidean space have unitary covariance that decreases exponentially with the distance. As we will see in Sec. 7.4, this feature has important implications for OOD detection.

THE RELATION OF INFINITE-WIDTH BNNS AND GPS. The connection between neural networks and GPs has recently gained significant attention. Neal [62] showed that a 1-hidden layer BNN converges to a GP in the infinite-width limit. More recently, the work by Lee *et al.* [184] and de G. Matthews *et al.* [185] extended this result to deeper networks, called *neural network Gaussian processes* (NNGP). Crucially, the kernel function of the related GP strictly depends on the used activation function. To better understand this connection we consider a fully connected network with L layers $l = 0, \ldots, L$ with width H_l. For each input we use $x^l(\boldsymbol{x})$ to represent the post-activation with $\boldsymbol{x}^0 = \boldsymbol{x}$ and $\boldsymbol{f}^l$ the pre-activation so that $f_i^l(\boldsymbol{x}) = b_i^{l-1} + \sum_{j=1}^{H_{l-1}} w_{ij}^{l-1} x_j^{l-1}$ with $x_j^l = h(f_j^l)$ and $h(\cdot)$ a point-wise activation function. Furthermore, the weights and biases are distributed according to $b_j^l \sim \mathcal{N}(0, \sigma_b^2)$ and $w_{ij}^l \sim \mathcal{N}(0, \sigma_w^2 / H_l)$. Given the independence of the weights and biases it follows that the post-activations are independent as well. Hence, the central limit theorem can be applied, and for $H_{l-1} \to \infty$ we obtain $f^l(\boldsymbol{x}) \sim \mathcal{GP}(0, C^l)$. The covariance C^l and therefore the prior

over functions is specified by the kernel induced by the network architecture [184]:

$$\begin{aligned} C^l(\boldsymbol{x}, \boldsymbol{x}') &= \overbrace{\mathbb{E}\left[f_i^l(\boldsymbol{x}) f_i^l(\boldsymbol{x}')\right]}^{k^l(\boldsymbol{x},\boldsymbol{x}')} - \overbrace{\mathbb{E}[f_i^l(\boldsymbol{x})]\mathbb{E}[f_i^l(\boldsymbol{x})]}^{=0} \\ &= \sigma_b^2 + \sigma_w^2 \mathbb{E}_{\left(f_i^{l-1}(\boldsymbol{x}), f_i^{l-1}(\boldsymbol{x}')\right) \sim \mathcal{GP}(0, K^{l-1})} \left[h\left(f_i^{l-1}(\boldsymbol{x})\right), h\left(f_i^{l-1}(\boldsymbol{x}')\right)\right] \end{aligned} \tag{7.4}$$

with K^l being the 2×2 covariance matrix formed by $k^l(\cdot, \cdot)$ using $\boldsymbol{x}$ and $\boldsymbol{x}'$. For the input layer, where no activation function is applied, the kernel is simply given by $k^0(\boldsymbol{x}, \boldsymbol{x}') = \sigma_b^2 + \sigma_w^2 \left(\frac{\boldsymbol{x}^T \boldsymbol{x}'}{n_{\text{in}}}\right)$. For the remaining layers, instead, the kernel expression is determined by the non-linearity. Some activation functions like sigmoid or hyperbolic tangent do not admit a known analytical form and therefore require a Monte Carlo estimate of Eq. 7.4 (cf. Fig. A.3). For others, Eq. 7.4 can be computed in closed form [e.g., 175, 186–189]. We list some kernel functions that are important for this study below, such as the one for ReLU networks [186]:

$$\begin{aligned} k_{\text{ReLU}}^l(\boldsymbol{x}, \boldsymbol{x}') &= \sigma_b^2 + \frac{\sigma_w^2}{2\pi} \sqrt{k^{l-1}(\boldsymbol{x}, \boldsymbol{x}) k^{l-1}(\boldsymbol{x}', \boldsymbol{x}')} \\ &\quad \cdot \left(\sin \theta_{\boldsymbol{x},\boldsymbol{x}'}^{l-1} + (\pi - \theta_{\boldsymbol{x},\boldsymbol{x}'}^{l-1}) \cos \theta_{\boldsymbol{x},\boldsymbol{x}'}^{l-1}\right) \\ \theta_{\boldsymbol{x},\boldsymbol{x}'}^l &= \cos^{-1}\left(\frac{k^l(\boldsymbol{x}, \boldsymbol{x}')}{\sqrt{k^l(\boldsymbol{x}, \boldsymbol{x}) k^l(\boldsymbol{x}', \boldsymbol{x}')}}\right) \end{aligned} \tag{7.5}$$

Interestingly, for some non-linearities a kernel that carries similar properties as the RBF in Eq. 7.3 can be induced by neural networks. This is the case for the cosine activation function which has the following closed form solution in the single hidden layer case [189]:

$$\begin{aligned} k_{\cos}^1(\boldsymbol{x}, \boldsymbol{x}') = \sigma_b^2 + \frac{\sigma_w^2}{2} \Bigg(&\exp\left(-\frac{\sigma_w^2 \|\boldsymbol{x} - \boldsymbol{x}'\|_2^2}{2d}\right) \\ &+ \exp\left(-\frac{\sigma_w^2 \|\boldsymbol{x} + \boldsymbol{x}'\|_2^2}{2d} - 2\sigma_b^2\right) \Bigg) \end{aligned} \tag{7.6}$$

Similarly, also the exponential activation function as deployed in RBF networks [190] leads to similar properties. This particular class of neural

networks defines computational units as linear combinations of radial basis functions $f(\boldsymbol{x}) = \sum_{j=1}^{H} w_j \exp\left(-\frac{1}{2\sigma_g^2}\|\boldsymbol{x} - \boldsymbol{\mu}_j\|^2\right) + b$. These networks, in the infinite-width case and when assuming a Gaussian prior over the centers $\boldsymbol{\mu}_j \sim \mathcal{N}(0, \sigma_\mu^2 \mathbb{I})$ and $w_j \sim \mathcal{N}(0, \sigma_w^2)$, also converges to a Gaussian process with an analytical kernel function:

$$k^1_{\text{rbfnet}}(\boldsymbol{x}, \boldsymbol{x}') = \sigma_b^2 + \sigma_w^2 \left(\frac{\sigma_e}{\sigma_\mu}\right)^d \exp\left(-\frac{\|\boldsymbol{x}\|^2}{2\sigma_m^2}\right) \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{x}'\|^2}{2\sigma_s^2}\right) \cdot \exp\left(-\frac{\|\boldsymbol{x}'\|^2}{2\sigma_m^2}\right) \tag{7.7}$$

where $1/\sigma_e^2 = 2/\sigma_g^2 + 1/\sigma_\mu^2$, $\sigma_s^2 = 2\sigma_g^2 + \sigma_g^4/\sigma_\mu^2$ and $\sigma_m^2 = 2\sigma_\mu^2 + \sigma_g^2$. The equivalence with the RBF kernel is explicit in the limit $\sigma_\mu^2 \to \infty$ [175].

7.3.3 *PAC-Bayes generalization bound*

Considering the supervised learning framework introduced in Chapter 2, the PAC theory [191] establishes a probabilistic bound on the generalization error of a given predictor. The theory is referred to as PAC-Bayes [192, 193] when a prior distribution p is defined to incorporate prior domain knowledge. Considering a distribution[2] q on the hypothesis $h \in \mathcal{H}$ and the space $\mathcal{B}(\mathcal{Y})$ of all conditional distributions $p(\boldsymbol{y} \mid h(\boldsymbol{x}))$, we can define a loss function $l : \mathcal{B}(\mathcal{Y}) \times \mathcal{Y} \to \mathbb{R}$. This gives rise to the empirical and true risk as $R_{\mathcal{D}}(q) := \frac{1}{N}\sum_{n=1}^{N} \mathbb{E}_{h\sim q}[l(p(\boldsymbol{y} \mid h(\boldsymbol{x}^{(n)})), \boldsymbol{y}^{(n)})]$ and $R(q) := \mathbb{E}_{(\boldsymbol{x}_\star, \boldsymbol{y}_\star)\sim p(\boldsymbol{x},\boldsymbol{y})}\mathbb{E}_{h\sim q}[l(p(\boldsymbol{y} \mid h(\boldsymbol{x}_\star)), \boldsymbol{y}_\star)]$, respectively, where $p(\boldsymbol{x}, \boldsymbol{y}) = p(\boldsymbol{x})p(\boldsymbol{y} \mid \boldsymbol{x})$. Note that when q is the Bayesian posterior as, for instance, in the GP case, predictions are made via the predictive posterior and it is thus more natural to consider the Bayes risk: $R_B(q) := \mathbb{E}_{(\boldsymbol{x}_\star, \boldsymbol{y}_\star)\sim p(\boldsymbol{x},\boldsymbol{y})}[l(\mathbb{E}_{h\sim q}[p(\boldsymbol{y} \mid h(\boldsymbol{x}_\star))], \boldsymbol{y}_\star)]$. It can be shown that $R_B(q) \leq 2R(q)$ for a quasi-convex loss [194, 195] and $R_B(q) \leq R(q)$ for a convex loss (using Jensen's inequality). Therefore, a bound over R also implies a bound over R_B. The PAC bound gives a probabilistic upper bound on the true risk $R(q)$ in terms of the empirical risk $R_{\mathcal{D}}(q)$ for a training set $\mathcal{D}$ as formulated in the following theorem:

Theorem 1 (PAC-Bayes theorem [193, 196]) *For any distribution $p(\boldsymbol{x}, \boldsymbol{y})$ over $\mathcal{X} \times \mathcal{Y}$ and bounded loss $l : \mathcal{B}(\mathcal{Y}) \times \mathcal{Y} \to [0, 1]$, for any prior p on the hypoth-*

2 Note that this distribution does not necessarily need to be the Bayesian posterior.

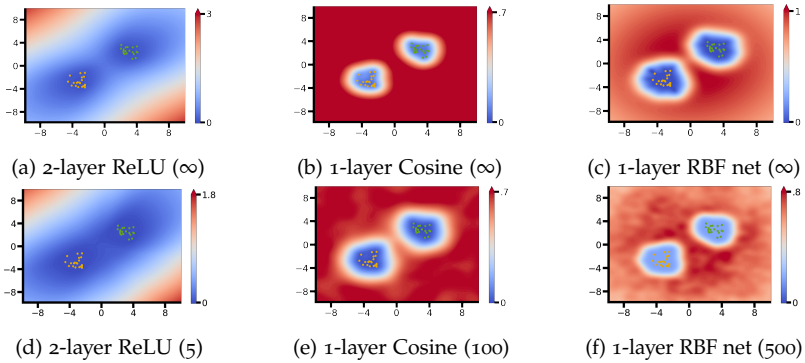
esis space $\mathcal{H}$ and any q that is absolutely continuous with respect to p, for any $\delta \in (0, 1]$ and $\beta > 0$ the following holds with probability at least $1 - \delta$ for a training set $\mathcal{D} \sim p(\boldsymbol{x}, \boldsymbol{y})$ of cardinality N:

$$\forall q : R(q) \leq \frac{1}{1 - e^{-\beta}} \left[1 - \exp\left(- \beta R_{\mathcal{D}}(q) - \frac{1}{N} \left(KL(q; p) + \log \frac{1}{\delta} \right) \right) \right] \quad . \tag{7.8}$$

For a fixed p, $\mathcal{D}$, δ, minimizing Eq. 7.8 is equivalent to minimizing $N\beta R_{\mathcal{D}}(q) + \mathrm{KL}(q; p)$. The requirement of a bounded loss function makes the use of the mean-squared error, and thus the negative log-likelihood, not applicable to this bound. Nevertheless, as long as the loss function measures the quality of predictions, the bound in Eq. 7.8 can be used to assess the generalization capabilities of a model. For this purpose, in our analysis we use as a surrogate loss the following [195]: $l_{\exp}(p(\boldsymbol{y} \mid h), \boldsymbol{y}) = 1 - \exp\left[- \frac{(\mathbb{E}_{\boldsymbol{y}}[p(\boldsymbol{y}|h)] - \boldsymbol{y})^2}{\sigma_l^2} \right]$, where we chose the hyperparameter $\sigma_l^2 = 1$. Interestingly, for small deviations we can take the first-order Taylor expansion and recover the MSE $l_{\exp}(\mathbb{E}_{\boldsymbol{y}}[p(\boldsymbol{y} \mid h)], \boldsymbol{y},) \approx (\mathbb{E}_{\boldsymbol{y}}[p(\boldsymbol{y} \mid h)] - \boldsymbol{y})^2 / \sigma_l^2$. This loss is quasi-convex and a bound on $R(q)$ below 0.5 thus induces a non-trivial bound on $R_B(q)$. While it is generally non-trivial to compute the KL in function space [197], the KL of a GP posterior from its prior has a closed-form solution [195].

7.4 THE ARCHITECTURE STRONGLY INFLUENCES OOD UNCERTAINTIES

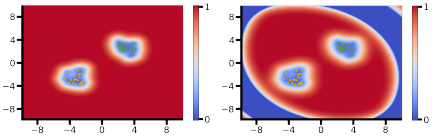
In the previous section, we recalled the connection between BNNs and GPs, namely that Bayesian inference in an infinite-width neural network can be studied in the GP framework (assuming a proper choice of weight-space prior). This connection allows studying how the kernels induced by architectural choices shape the prior in function space, and how these choices ultimately determine OOD behavior. In this section, we analyze this OOD behavior for traditional as well as NNGP-induced kernels. To minimize the impact of the approximations on our results, we exclusively focus on conjugate settings, i.e., regression (see SM A.3.1 for classification results). Furthermore, this choice of Gaussian likelihood induces a direct correspondence between function values and outcomes up to noise corruptions. Hence, prior knowledge about outcomes can be encoded in function space through the choice of a meaningful function space prior.



(a) 2-layer ReLU (∞) (b) 1-layer Cosine (∞) (c) 1-layer RBF net (∞)

(d) 2-layer ReLU (5) (e) 1-layer Cosine (100) (f) 1-layer RBF net (500)

FIGURE 7.2: **Standard deviation $\sigma(\boldsymbol{f}_*)$ of the predictive posterior of BNNs.** We perform Bayesian inference on a mixture of two Gaussians dataset considering different priors in function space induced by different architectural choices. The problem is treated as regression task to allow exact inference in combination with GPs **(a, b, c)**. Predictive uncertainties for finite-width networks are obtained using HMC (**d, e, f**).



(a) RBF Kernel (b) Periodic Kernel

FIGURE 7.3: **Standard deviation $\sigma(\boldsymbol{f}_*)$ of the predictive posterior using GPs with common kernel functions.** GP regression is performed on the same dataset as in Fig. 7.2.

UNCERTAINTY QUANTIFICATION FOR OOD DETECTION. Uncertainty can be quantified in multiple ways, but is often measured as the entropy of the predictive posterior. The predictive posterior, however, captures both aleatoric and epistemic uncertainty, which does not allow a distinction between OOD and ambiguous inputs [173]. While our choice of likelihood does not permit the modelling of input-dependent uncertainty (Gaussian with fixed variance), a softmax classifier can capture heteroscedastic aleatoric uncertainty arbitrarily well by outputting an input-dependent categorical distribution.[3] For this reason, uncertainty should be quantified in a way that allows OOD detection to be based on epistemic uncertainty only. The function space view of GPs naturally provides such measure of epistemic uncertainty by considering the standard deviation of the posterior over function values $\sigma(\boldsymbol{f}_*) \equiv \sqrt{\operatorname{diag}(\mathrm{C}(\boldsymbol{f}_*))}$ (illustrated in

3 There are also expressive likelihood choices to capture heteroscedastic uncertainty for continuous variables [e.g., 198].

Fig. 7.1a). Such measure can be naturally translated to BNNs by looking at the disagreement between network outputs when sampling from $p(\boldsymbol{w} \mid \mathcal{D})$. Note that in classification tasks all models drawn from $p(\boldsymbol{w} \mid \mathcal{D})$ might lead to a high-entropy softmax without disagreement, and therefore only an uncertainty measure based on model disagreement prevents one from misjudging ambiguous points as OOD. As OOD detection is the focus of this paper, we always quantify uncertainty in terms of model disagreement (see Sec. 3.4 for more details).

BAYESIAN STATISTICS AND OOD DETECTION HAVE NO INTRINSIC CONNECTION. As conceptually illustrated in Fig. 7.1b and demonstrated using exact Bayesian inference in Fig. 7.3b, predictive uncertainties are not necessarily reflective of $p(\boldsymbol{x})$ (and thus not suited for OOD detection), irrespective of how well the underlying task is solved.

GP REGRESSION WITH AN RBF KERNEL. We next examine the predictive posterior of a GP with squared exponential (RBF) kernel (Eq. 7.3). Fig. 7.3a shows epistemic uncertainty as the standard deviation of $p(\boldsymbol{f}_* \mid X_*, X, \boldsymbol{y})$. We can notice that $\sigma(\boldsymbol{f}_*)$ nicely captures the data manifold and is thus well suited for OOD detection. This behavior can be understood by considering Eq. 7.2 while noting that $k(\boldsymbol{x}, \boldsymbol{x}') = \text{const}$ if $\boldsymbol{x} = \boldsymbol{x}'$, and that the variance of posterior function values can be written as $\sigma^2(\boldsymbol{f}_*) = k(\boldsymbol{x}_*, \boldsymbol{x}_*) - \sum_{i=1}^{n} \beta_i(\boldsymbol{x}_*) k(\boldsymbol{x}_*, \boldsymbol{x}_i)$ (cf. SM A.2), where β_i are dataset- and input-dependent. The second term is reminiscent of using kernel density estimation (KDE) to approximate $p(\boldsymbol{x})$, applying a Gaussian kernel, while the first term is the (constant) prior variance. Hence, the link between Bayesian inference and OOD detection can be made explicit, as the posterior variance is inversely related to the input distribution.

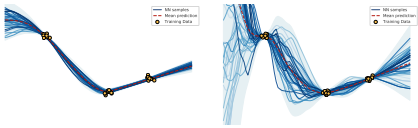
Loosely speaking, in GP regression as in Eq. 7.2 and kernels where the KDE analogy holds, epistemic uncertainty can roughly be described as $\text{const} - p(\boldsymbol{x})$.[4] In this view, one starts with high (prior) uncertainty everywhere, which is only reduced where data is observed. By contrast, learning a normalized generative model (e.g., using normalizing flows [171]) often requires to start from an arbitrary probability distribution.

THE OOD BEHAVIOR INDUCED BY NNGP KERNELS. Fig. 7.2a illustrates $\sigma(\boldsymbol{f}_*)$ for an infinite-width 2-layer ReLU network (see Fig. A.2 for other

4 Note, the KDE approximation of $p(\boldsymbol{x})$ is likely to deteriorate massively if the dimensionality of $\boldsymbol{x}$ increases.

common architectural choices). It is already visually apparent, that in this case the kernel is less suited for OOD detection compared to an RBF kernel. Moreover, we cannot justify why OOD detection based on $\sigma(\boldsymbol{f}_*)$ would be principled for this kernel as the KDE analogy does not hold. This is due to two reasons: (1) the prior uncertainty $k(\boldsymbol{x}_*, \boldsymbol{x}_*)$ is not constant (Fig. A.1), and (2) we empirically do not observe that $k(\boldsymbol{x}, \boldsymbol{x}')$ can be related to a distance measure (e. g., Fig. A.4). We therefore argue that more theoretical work is necessary if one aims to justify uncertainty-based OOD detection with common architectural choices from the perspective of NNGP kernels. On this note, maintaining parameter uncertainty and being able to detect OOD samples are often considered crucial requirements of systems deployed in safety-critical applications. Given that Bayesian inference is not intrinsically linked to OOD detection, care should be taken to precisely communicate safety-relevant capabilities of BNNs to practitioners.

However, the NNGP perspective also allows us to choose an architecture such that the BNN's uncertainty resembles the, for these problems, desirable OOD behavior of the GP with RBF kernel described above. In particular, the cosine (Eq. 7.6) and RBF (Eq. 7.7) network induce kernels that are related to the RBF kernel. The last term in Eq. 7.6 quickly converges to zero for moderate norms of $\boldsymbol{x}$ or $\boldsymbol{x}'$ (or high σ_b^2), which explains why the uncertainty behavior of Fig. 7.2b is qualitatively identical to Fig. 7.3a. In case of the RBF network, the RBF kernel is recovered if $\|\boldsymbol{x}\|$ and $\|\boldsymbol{x}'\|$ are small compared to σ_m^2, explaining why the uncertainty faints towards the boundaries of Fig. 7.2c. These examples show that in the non-parametric limit and for (low-dimensional) regression tasks, BNNs can be constructed such that uncertainty-based OOD detection can be justified through mathematical argumentation. We next study whether the observations made in the infinite-width limit are relevant for studying finite-width neural networks.

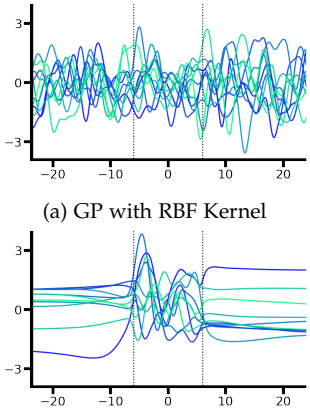


(a) Width-aware prior (b) Standard prior

FIGURE 7.4: **The importance of the choice of weight prior $p(\boldsymbol{w})$.** Here, we perform 1d regression using HMC with either **(a)** a width-aware prior $\mathcal{N}(0, \frac{\sigma_w^2}{H_l})$ or **(b)** a standard prior $\mathcal{N}(0, \sigma_w^2)$.

INFINITE-WIDTH UNCERTAINTY IS CONSISTENT WITH THE FINITE-WIDTH UNCERTAINTY. For finite-width BNNs exact Bayesian inference is intractable.

To mitigate the effects of approximate inference we resort to Hamiltonian Monte Carlo [HMC, 90, 91]. Fig. 7.2d to 7.2f show an estimate of $\sigma(\boldsymbol{f}_*)$ for finite-width networks corresponding to the non-parametric limits studied above (illustrations with another dataset can be found in Fig. A.5). Already for moderate layer widths, the modelled uncertainty resembles the one of the corresponding NNGP. Given this close correspondence, we conjecture that the tools available for the infinite-width case are useful for designing architectural guidelines that enhance OOD detection. We illustrated this on low-dimensional problems by studying desirable function space properties induced by the RBF kernel, which can be translated to BNN architectures.



(a) GP with RBF Kernel

(b) Network with Ridgelet Prior

FIGURE 7.5: **OOD challenges when encoding function space properties into weight space prior. (a)** Samples from a GP prior $p(\boldsymbol{f} \mid X)$. **(b)** Samples from a 1-layer Tanh network (width: 3000) using the Ridgelet prior [66] corresponding to the GP in (a). Dotted lines denote the domain $\mathcal{X}_R$ within which the Ridgelet prior was matched to the target GP.

7.5 THE CHOICE OF WEIGHT SPACE PRIOR MATTERS FOR OOD DETECTION

For a neural network, the prior in function space is induced by the architecture and the prior in weight space [67].

In the previous section, we restricted ourselves to a particular class of weight space priors which allowed us to study the architectural choices in the infinite-width limit. While, in practice, a wide variety of weight space prior choices might lead to good generalization (with respect to test data from $p(\boldsymbol{x})p(\boldsymbol{y} \mid \boldsymbol{x})$, e.g., cf. work by Izmailov *et al.* [92]), the uncertainty behavior that is induced out-of-distribution might vary drastically. This is illustrated in Fig. 7.4, where for the same network two different choices of $p(\boldsymbol{w})$ lead to vastly different predictive uncertainties despite the fact that both choices fit the data well.

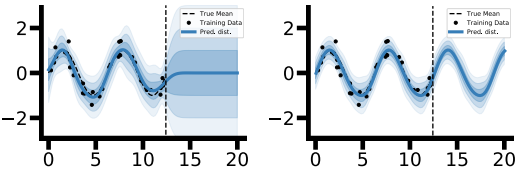
	$R_{\mathcal{D}}(q)$	$R_{B,\mathcal{D}}(q)$	$\mathrm{KL}(q,p)$	PAC	$\log p(\mathcal{D})$	Test MSE
RBF	$0.111^{\pm.007}$	$0.048^{\pm.006}$	$8.849^{\pm.119}$	$0.496^{\pm.010}$	$-27.595^{\pm.549}$	$0.029^{\pm.016}$
ESS	$0.094^{\pm.006}$	$0.055^{\pm.006}$	$6.143^{\pm.096}$	$0.419^{\pm.008}$	$-23.532^{\pm.406}$	$0.014^{\pm.008}$

TABLE 7.1: Empirical risk $R_{\mathcal{D}}(q)$, empirical version of the Bayes risk $R_{B,\mathcal{D}}(q)$, KL divergence, PAC bound ($\delta = 0.05$, $\beta = 2$), log marginal likelihood and mean-squared error (MSE) on a withheld test set for the examples shown in Fig. 7.6 (cf. Sec. 7.3.3).

In Sec. 7.4, we use the infinite-width limit to obtain a function space view that allows to make interesting conjectures about predictive uncertainties on OOD data. Recently, multiple studies suggested ways to either explicitly encode function space properties into the weight space prior $p(\boldsymbol{w})$ or to use a function space prior when performing approximate inference in neural networks [66, 199–201]. However, all these methods depend on the specification of a set $\mathcal{X}_R \subseteq \mathcal{X}$ on which desired function space properties will be enforced (Fig. 7.5). For instance, the Ridgelet prior proposed by Matsubara, Oates & Briol [66] provides an asymptotically correct weight space prior construction that induces a given GP prior. Thus, on $\mathcal{X}_R$ this method allows to meaningfully encode prior knowledge to guide Bayesian inference. But, an a priori specification of a set $\mathcal{X}_R$ which can be largely covered by samples $\boldsymbol{x}_R \in \mathcal{X}_R$ to ensure the desired prior specification on $\mathcal{X}_R$ is practically challenging, and conceptually related to the idea of training on OOD data to calibrate respective uncertainties (cf. Sec. 7.2). Note, we do not aim to phrase this OOD effect as a drawback of these methods, as we do not consider Bayesian inference to be intrinsically related to OOD detectors. It is, however, important to keep in mind that function space properties deemed beneficial for OOD detection are not straightforward to induce when working in weight space.

7.6 A TRADE-OFF BETWEEN GENERALIZATION AND OOD DETECTION

An important desideratum that modelers attempt to achieve when applying Bayesian statistics is good generalization through the incorporation of relevant prior knowledge. Is this desideratum generally in conflict with having high uncertainty on OOD data? In this section, we provide arguments indicating that the answer to this question can be *yes*. Consider data with known periodic structure inside the support of $p(\boldsymbol{x})$ (Fig. 7.6).



(a) GP with RBF Kernel (b) GP with ESS kernel

FIGURE 7.6: **Generalization and OOD detection.** Mean and the first three standard deviations of the predictive posterior for different function space priors.

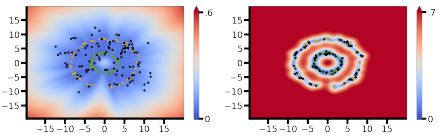
We can either choose to ignore our prior knowledge by selecting a GP prior with RBF kernel (Fig. 7.6a), or we explicitly incorporate domain knowledge using a function space prior that allocates its mass on periodic functions e.g. the exp-sine-squared (ESS) kernel.[5] In the former case, we know that OOD uncertainties will be useful for OOD detection (Sec. 7.4). In the latter case, however, we see that uncertainties do not reflect $p(\boldsymbol{x})$. By contrast, the roles are reversed when it comes to assessing generalization.

Here, the choice of a periodic kernel has clear benefits, as we can see visually and by comparing, for instance, PAC-Bayes generalization bounds (reviewed in Sec. 7.3.3) which can be tightened by minimizing $N\beta R_{\mathcal{D}}\left(p(\boldsymbol{f} \mid \mathcal{D})\right) + \mathrm{KL}\left(p(\boldsymbol{f} \mid \mathcal{D}); p(\boldsymbol{f})\right)$, where β is a hyperparameter and $R_{\mathcal{D}}\left(p(\boldsymbol{f} \mid \mathcal{D})\right)$ denotes the empirical risk. As reported in Table 7.1, the bound's value for the ESS kernel is lower than the RBF. Interestingly, the bound is minimized if both terms are minimized, and the second term explicitly asks the posterior to remain close to the prior. Therefore, the second term benefits from restricting the prior function space through the incorporation of prior knowledge, counteracting the need of a rich function space for inducing high OOD uncertainties (Fig. 7.1). Consistent with this, we observe in Table 7.1 that for the ESS kernel the log marginal likelihood $\log p(\mathcal{D})$, which is often considered as a model selection criterion [57], is higher and the test error (on a withheld test set) is lower.

An orthogonal problem often considered in the literature is generalization under dataset (or covariate) shift [49, 68, 202]. In this case, one deliberately seeks to provide meaningful predictions on test data $p_{\text{test}}(\boldsymbol{x})$ that might not overlap in support with our training input distribution $p(\boldsymbol{x})$. Therefore, prior knowledge needs to explicitly encode how to obtain "generalization on OOD data" as the data cannot speak for themselves. Such priors are proposed by Izmailov *et al.* [202], but also Fig. 7.6b can be viewed as an example of how prior encoding specifies how to generalize to OOD data.

5 Note, that such incorporation of prior knowledge is often done in a way that is agnostic to the unknown support of $p(\boldsymbol{x})$.

7.7 ON THE PRACTICAL VALIDATION OF OOD PROPERTIES



(a) 1-layer ReLU (∞) (b) 1-layer Cosine (∞)

FIGURE 7.7: **Sampling from regions of low uncertainty.** Here, we treat epistemic uncertainty (measured as $\sigma(\boldsymbol{f}_*)$) as an energy function and perform rejection sampling (black dots). If uncertainty faithfully captures $p(\boldsymbol{x})$, these samples should resemble in-distribution data.

Relating the epistemic uncertainty induced by a BNN to $p(\boldsymbol{x})$ opens up interesting new possibilities. As we show in SM A.2, epistemic uncertainty of a GP with RBF kernel can be related to a KDE approximation of $p(\boldsymbol{x})$. Therefore, one may consider the epistemic uncertainty as an energy function to create a generative model from which to sample from the input space (Fig. 7.7). Moreover, this approach may be used to empirically validate the OOD capabilities of a BNN. Indeed, OOD performance is commonly validated by selecting a specific set of known OOD datasets [92]. However, as the OOD space comprises everything except the in-distribution data, it is infeasible to gain good coverage on high-dimensional data with such testing approach. We therefore suggest a reverse approach that checks the consistency in regions of certainty instead of uncertain ones by sampling via the epistemic uncertainty. Indeed, if these samples and thus the generative model based on uncertainty estimates are consistent with the in-distribution data then a strong indication for trustworthiness is provided.

Additionally, a discrepancy measure [203, 204] can be used to compare the generated samples and training data points and quantitatively assess to which extent the epistemic uncertainty is related to $p(\boldsymbol{x})$. Moreover, this approach can open up new research opportunities. For instance, a continual learner may use its uncertainty to generate its own replay data to combat forgetting (cf. SM A.3.3).

7.8 CONCLUSION

In this chapter, we challenge the common view that uncertainty-based OOD detection with BNNs is intrinsically justified. Our arguments are almost exclusively based on low-dimensional problems and cannot easily be generalized to real-world problems, but our observations are consistent with the fact that empirically BNNs are often only marginally superior

in OOD detection compared to models that do not maintain epistemic uncertainty [40, 49]. Overall, this work highlights fundamental limitations of BNNs for OOD detection that are not solely explained by the use of approximate inference.

8

DISCUSSION

In this thesis, we research two aspects, uncertainty and continual learning, that we assume to be cornerstones of natural intelligence, and are thus essential to sustained progress in the field of artificial intelligence. As outlined in Chapter 2, we restrict ourselves to supervised learning in feedforward neural networks,[1] a setup that is of increasing practical relevance in many real-world applications [e. g., 19, 206].

However, as discussed in Chapter 3, the common practice of estimating a single set of parameters from limited data for the purpose of approximating an unknown data generative process fails to capture epistemic uncertainty, i. e., the ability to *know what we don't know*. Importantly, and as opposed to memorization, learning requires prior knowledge [69], even if just in the form of heuristically-evolved inductive biases [67]. A way to incorporate prior knowledge in a principled manner is through Bayesian statistics, which allows the representation of uncertainties by treating learning as a probabilistic inference problem (cf. Sec. 3.2), while still relying on modelling assumptions that in practice can drastically alter the meaning of uncertainties (as we discuss in Ref. [1]). Yet, applying this Bayesian framework to neural networks is non-trivial for two major reasons. The first is the need to resort to approximate inference (cf. Sec. 3.5), which can yield vastly different results as shown in Chapter 6. The second is our current inability to formulate and encode meaningful prior knowledge (cf. Chapter 7). Knowing what we don't know depends on what we know a priori, and thus despite seeing the same evidence, two priors can yield predictive distributions that perform vastly different in terms of generalization or uncertainty estimation (cf. Sec. 3.3). Also humans exhibit different levels of self-confidence [207], which may indicate that genetically and environmentally formed prior knowledge shapes uncertainties. Yet, uncertainties are still considered to be useful for biological agents in an ever changing world [208] and provide an explicit incentive to gather information for uncertainty reduction [209]. Therefore, incorporating knowledge uncertainties may turn out useful for artificial intelligence as well, even though more research to facilitate this goal is needed.

1 But see our study on CL in RNNs in Ref. [33] and, for instance, the work by Fortunato, Blundell & Vinyals [205] on how to perform approximate Bayesian inference in RNNs.

The second main topic of this thesis is continual or lifelong learning. In biology, there is no training and subsequent testing phase. Instead, humans can continually learn to adapt to their environment and to acquire new skills without (catastrophically) forgetting existing ones. Unfortunately, standard training procedures for neural networks, which are of immense practical relevance nowadays [e. g. 97], do not allow for this ability (cf. Chapter 4).

After introducing and reviewing both main topics in Chapters 3 and 4, we presented our hypernetwork-based method to continually learn a sequence of tasks in Chapter 5. This method entangles task-specific solutions into a single shared meta-model, the hypernetwork, and thus requires explicit task-inference (cf. Sec. 4.1.3). This setup shifts the problem of continual learning to the meta-level where forgetting can be explicitly addressed via a simple regularization technique (cf. Eq. 5.3 and Eq. 6.1), which we empirically demonstrated to be able to prevent forgetting successfully even for long task sequences or challenging computer vision tasks [39, 40]. Task-inference, on the other hand, remains a challenging problem. We propose replay-based task-inference as well as uncertainty-based task-inference. Replay-based task-inference (e. g., HNET+TIR, Sec. 5.2.3) relies on generative modelling, and excels on tasks such as SplitMNIST (cf. Fig. 4.1) where a generative model is easy to obtain. However, apart from scalability concerns, it is conceptually questionable whether generative modelling should be employed for learning a discriminative model continually.

Therefore, we also studied uncertainty-based task-inference, which already yields substantial performance improvements compared to prior-focused methods (cf. Sec. 4.1.2.1) when using uncalibrated out-of-distribution (OOD) uncertainties of deterministic classifiers (cf. Table 5.1). Fortunately, the hypernetwork-based approach from Chapter 5 can be extended to capture task-specific parameter uncertainty as we present in Chapter 6. This probabilistic extension allows us to reflect epistemic uncertainties while gaining more robust task-inference performance. While this approach is computationally more expensive (as to be expected from an instantiation of Bayesian neural networks, Sec. 3.5), it shows consistent performance improvements and is equipped with multiple conceptual advances that come with the Bayesian treatment, including the ability to integrate new evidence within tasks in a mathematically sound way (cf. Eq. 4.5) as well as the ability to specify task-specific prior knowledge for explicit knowledge transfer. Yet, despite setting state-of-the-art performance on benchmarks such as SplitCIFAR, task-inference proved to be a major bottleneck in this framework (cf. Table 6.3). To understand the reasons for this, we studied

uncertainty-based OOD detection in Chapter 7, which is required for a naive use of uncertainties for task-inference.

Unfortunately, Bayesian neural networks often fall short of expectations in their ability to detect OOD inputs (cf. Sec. 7.1). However, given the arbitrariness of prior selection in Bayesian deep learning (cf. Sec. 3.5) as well as the challenges arising due to the use of approximate inference (e. g., cf. Sec. 4.1.2.1), it should not come as a surprise that uncertainties do not reflect our intuitive desiderata.[2] Even under exact inference, it is currently unknown how to specify prior knowledge in real-world problems such that posterior uncertainties reflect the data-generating process in the sense that all OOD inputs are associated with high uncertainty (cf. Chapter 7). Furthermore, it remains to be discussed by the community whether OOD detection should even be an important desideratum for Bayesian discriminative models. We argue, that it may be preferable to focus on finding priors that exhibit good generalization (including generalization to OOD data, Sec. 7.6), and highlight that solely requiring good OOD detection may be equivalent to generative modelling (cf. Sec. 7.7 and SM A.3.3). Note, if OOD is defined as in Sec. 7.2, i. e., as inputs not sampled from $p(\boldsymbol{x})$, generative modelling can be employed directly for OOD detection [210, 211]. Also in this case, Bayesian statistics can be useful for dealing with the intricacies associated with density estimation on high-dimensional real-world data [212].

While boosting the performance of naive uncertainty-based task-inference, as mainly studied in Chapter 6, might heavily depend on progress in prior engineering, it should be stressed that the problem of task-inference is conceptually simpler than the problem of out-of-distribution detection. In particular, the coreset-based approach suggested in Sec. 6.3 allows to explicitly calibrate uncertainties for task-inference without violating CL desiderata. However, this approach relies on updating posteriors using the recursive Bayesian update, which strongly suffers from the use of approximate inference (cf. Sec. 4.1.2.1). Thus, in summary, progress in either the engineering of prior distributions or the improvement of methods for approximate inference will automatically translate into performance gains for our suggested *posterior meta-replay* framework.

Overall, we provide a multitude of contributions to the respective fields of deep learning research, ranging from (i) exploring methods for approximate inference [44], over (ii) advancing continual learning [33, 39, 40], and (iii) improving our understanding of the role of posterior-induced uncertainties

2 These desiderata might even be conflicting as outlined in Sec. 7.6.

for OOD detection [95, 116], to (iv) assessing the role of model misspecification in neural network regression [1]. These are important and still largely unresolved challenges, but challenges that receive increasing research attention and have seen steady progress in recent years. We thus look into the future full of expectation and assume to see continuing progress that ultimately leads to practical and trustworthy applications of AI.

A

SUPPLEMENTARY MATERIAL: UNCERTAINTY-BASED OUT-OF-DISTRIBUTION DETECTION REQUIRES SUITABLE FUNCTION SPACE PRIORS

This chapter's content is taken from an online preprint ***authored by****: Francesco D'Angelo* and Christian Henning* [95].*

** These authors contributed equally.*

A.1 WHAT IS AN OUT-OF-DISTRIBUTION INPUT?

Pimentel *et al.* [182] reviews methods for outlier detection, putting them coarsely into five categories: (1) probabilistic, (2) distance-based, (3) reconstruction-based, (4) domain-based, and (5) information-theoretic. Our focus lies on a probabilistic characterization of an OOD point, where a statistical criterion allows to decide whether a given input is significantly different from the observed training population. In this section, we discuss pitfalls regarding obvious choices for such a criterion in order to highlight the difficulties that arise when attempting to agree on a single (or application-dependent) mathematical definition of outliers.

There are many possible definitions that could be considered for a point to be OOD as we will outline below. Any of these definitions may change the notion of OOD and will therefore affect how a BNN should be designed such that predictive uncertainty adheres to the underlying OOD definition. Considering a generative process $p(\boldsymbol{x})$, the first question arising is regarding the regions outside the support of $p(\boldsymbol{x})$ or even the space outside the manifold where samples $\boldsymbol{x}$ are defined on. For instance, assume the data to be images embedded on a lower-dimensional manifold. Are points outside this manifold clearly OOD, given that minor noise corruptions are likely to leave the manifold? Even disregarding these topological issues solely focusing on the density $p(\boldsymbol{x})$, makes the distinction between in- and out-of-distribution challenging. For instance, a threshold-criterion on the density might cause samples in a zero probability region to be considered in-distribution [126]. To overcome such challenges, one may resort to concepts from information theory, such as the notion of a typical set [53, 210]. Unfortunately, an OOD criterion based on this notion would require looking at sets rather than

individual points. We hope that this short outline highlights the challenges regarding the definition of OOD, but also clarifies that a proper definition is relevant when assessing OOD capabilities on high-dimensional data, where a visual assessment as in Fig. 7.2 is not possible.

A.2 ON THE RELATION BETWEEN GP REGRESSION AND KERNEL DENSITY ESTIMATION

GP regression with an RBF kernel allows a direct understanding of the OOD capabilities that a Bayesian posterior may possess, as the a priori uniform epistemic uncertainty is reduced in direct correspondence to density of $p(\boldsymbol{x})$.

Below, we rewrite the variance of an input $\boldsymbol{x}_*$ as defined in Eq. 7.2:

$$\sigma^2(\boldsymbol{f}_*) = k(\boldsymbol{x}_*, \boldsymbol{x}_*) - \sum_{i=1}^{n} \beta_i(\boldsymbol{x}_*) k(\boldsymbol{x}_*, \boldsymbol{x}_i) \tag{A.1}$$

with $\beta_i(\boldsymbol{x}_*) = \sum_{j=1}^{n} \left(K(X,X) + \sigma^2 \mathbb{I}\right)^{-1}_{ij} k(\boldsymbol{x}_*, \boldsymbol{x}_j)$.

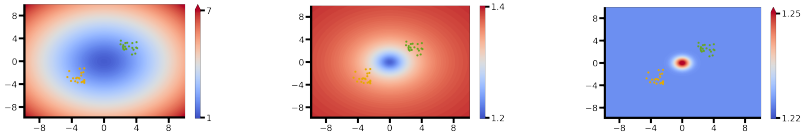
Note, that $k(\boldsymbol{x}_*, \boldsymbol{x}_*)$ is a positive constant for an RBF kernel, and that $\beta_i(\boldsymbol{x}_*) \geq 0$. Furthermore, as the RBF kernel $k(\boldsymbol{x}_*, \boldsymbol{x}_j)$ behaves exponentially inverse to the distance between $\boldsymbol{x}_*$ and $\boldsymbol{x}_j$, $\beta_i(\boldsymbol{x}_*)$ is approximately zero for all $\boldsymbol{x}_*$ that are far from all training points. In addition, a training point $\boldsymbol{x}_i$ can only decrease the prior variance if it is close to $\boldsymbol{x}_*$. This analogy closely resembles the philosophy of KDE, which becomes an exact generative model in the limit of infinite data and bandwidth $l \to 0$.

A.3 ADDITIONAL EXPERIMENTS AND RESULTS

In this section, we report additional experiments and results.

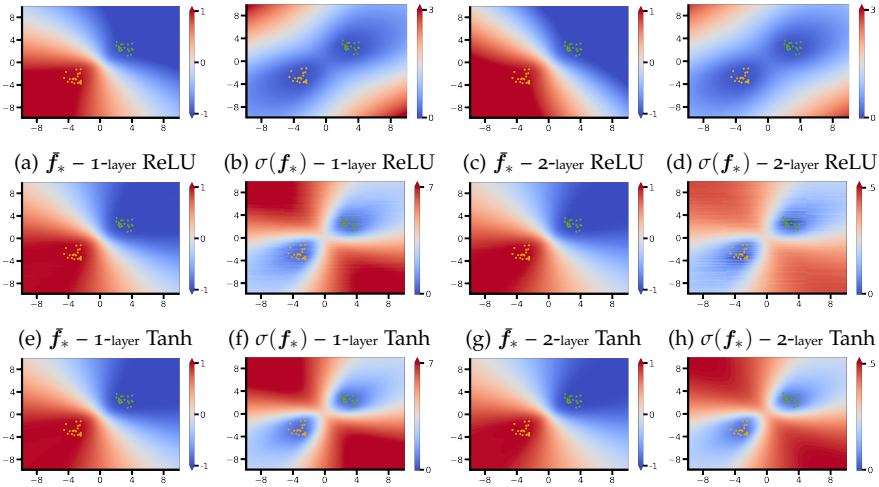
Fig. A.1 shows the prior's standard deviation over function values $\sqrt{k(\boldsymbol{x}_*, \boldsymbol{x}_*)}$ for several NNGP kernels. Note, a kernel that a priori treats locations $\boldsymbol{x}_*$ differently might not be desirable for OOD detection as the data's influence on posterior uncertainties might be hard to interpret (cf. Sec. 7.4).

Fig. A.2 shows GP regression results with the GMM dataset (Sec. A.4) for several NNGP kernels. The plots show that the underlying task can be solved well with all considered architectures (as indicated by the predictive mean $\bar{\boldsymbol{f}}_*$ that captures the ground-truth targets in-distribution), even though the uncertainty behavior OOD is vastly different and not reflective of $p(\boldsymbol{x})$.



(a) 1-layer ReLU (b) 1-layer Erf (c) 1-layer Cosine

FIGURE A.1: **NNGP kernel values $\sqrt{k(\boldsymbol{x}_*, \boldsymbol{x}_*)}$ for various architectural choices.** Note, that the NNGP kernel value $k(\boldsymbol{x}_*, \boldsymbol{x}_*)$ represents the prior variance of function values under the induced GP prior at the location $\boldsymbol{x}_*$ (cf. Eq. 7.1). As emphasized in Sec. 7.4, $k(\boldsymbol{x}_*, \boldsymbol{x}_*)$ is constant for an RBF kernel, which has important implications for OOD detection, such as that a priori (before seeing any data) all points are treated equally. This is not the case for the ReLU kernel, which has an angular dependence and depends on the input norm (cf. Eq. 7.5 and its dependence on $k^0(\boldsymbol{x}, \boldsymbol{x}')$). The kernel induced by networks using an error function (Erf) or a cosine as nonlinearity seems to be more desirable in this respect (note the scale of the colorbars).



(a) $\bar{\boldsymbol{f}}_*$ – 1-layer ReLU (b) $\sigma(\boldsymbol{f}_*)$ – 1-layer ReLU (c) $\bar{\boldsymbol{f}}_*$ – 2-layer ReLU (d) $\sigma(\boldsymbol{f}_*)$ – 2-layer ReLU

(e) $\bar{\boldsymbol{f}}_*$ – 1-layer Tanh (f) $\sigma(\boldsymbol{f}_*)$ – 1-layer Tanh (g) $\bar{\boldsymbol{f}}_*$ – 2-layer Tanh (h) $\sigma(\boldsymbol{f}_*)$ – 2-layer Tanh

(i) $\bar{\boldsymbol{f}}_*$ – 1-layer Erf (j) $\sigma(\boldsymbol{f}_*)$ – 1-layer Erf (k) $\bar{\boldsymbol{f}}_*$ – 2-layer Erf (l) $\sigma(\boldsymbol{f}_*)$ – 2-layer Erf

FIGURE A.2: **Mean $\bar{\boldsymbol{f}}_*$ and standard deviation $\sigma(\boldsymbol{f}_*)$ of the posterior $p(\boldsymbol{f}_* \mid X_*, X, \boldsymbol{y})$ for GP regression with NNGP kernels** for various architectural choices, such as number of layers or non-linearity. Note, that Tanh and Erf nonlinearities are quite similar in shape, which is reflected in the similar predictive posterior that is induced by these networks. We use the analytically known kernel expression for the Erf kernel [175], and use MC sampling for the Tanh network (cf. Fig. A.3).

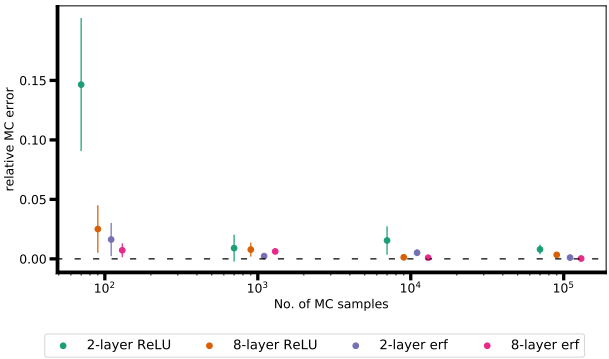


FIGURE A.3: **Monte Carlo error when estimating NNGP kernel values.** Eq. 7.4 requires estimation whenever no analytic kernel expression is available (for instance, when using a hyberbolic tangent nonlinearity as in Fig. A.2). Here, we visualize the error caused by this approximation for ReLU and error function (erf) networks when computing kernel values $k(\boldsymbol{x}_*, \boldsymbol{x}_*)$. Eq. 7.4 requires a recursive estimation of expected values, where we estimate each of them using N samples ($N \in [10^2, 10^3, 10^4, 10^5]$). Note, that even small errors can cause eigenvalues of the kernel matrix to become negative. However, with our chosen likelihood variance of $\sigma = 0.02$ we experience no numerical instabilities during inference, and obtain consistent results using either analytic or estimated ($N = 10^5$) kernel matrices.

Fig. A.3 (in combination with Fig. A.2) highlights that also approximated kernel values can be used to study architectures with no known closed-form solution for Eq. 7.4, as the posterior seems to be only marginally effected by the MC estimation.

Fig. A.4 visualizes that NNGP kernels for common architectural choices do not encode for Euclidean distances. Kernels that monotonically decrease with increasing distance are, however, important to apply the KDE interpretation of SM A.2.

Finally, in Fig. A.5 we investigate a more challenging dataset with two concentric rings. Note, that the center region as well as the region between the two rings can be considered OOD (not included in the support of $p(\boldsymbol{x})$). A good uncertainty-based OOD detector should therefore depict high uncertainties in those regions, which is not the case for ReLU networks.

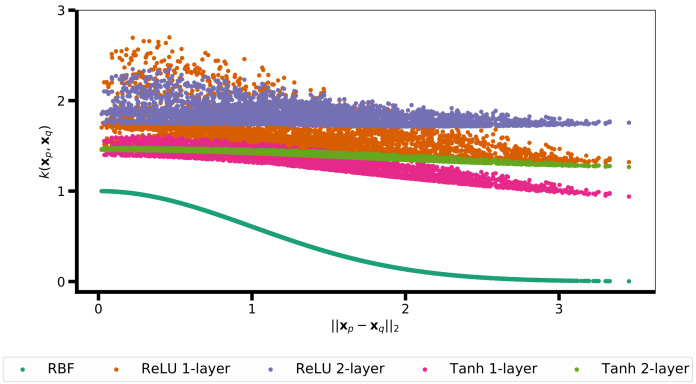
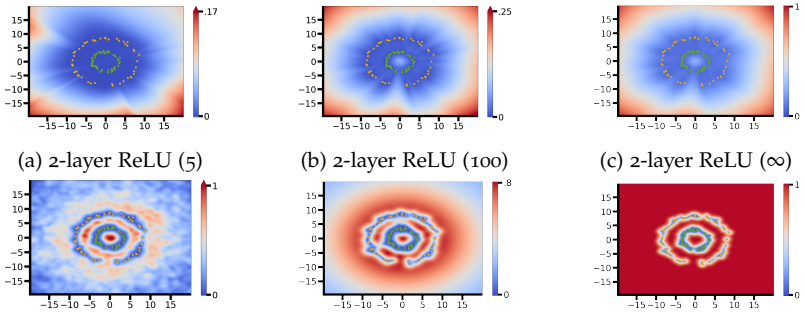


FIGURE A.4: **NNGP kernel values do not generally reflect Euclidean distances.** This figure shows kernel values $k(\boldsymbol{x}_q, \boldsymbol{x}_p)$ plotted as a function of the Euclidean distance $\|\boldsymbol{x}_q - \boldsymbol{x}_p\|_2$ (using pairs of training points from the two Gaussian mixtures dataset). As outlined in Sec. 7.4 and SM A.2, the interpretation of an RBF kernel as Gaussian kernel that can be used in a KDE of $p(\boldsymbol{x})$ is important to justify implied OOD, at least for low-dimensional problems. Unfortunately, the kernels induced by common architectures do not seem to be distance-aware and are thus not useful for KDE.



(a) 2-layer ReLU (5) (b) 2-layer ReLU (100) (c) 2-layer ReLU (∞)

(d) RBF net (500) (e) $\sigma(\boldsymbol{f}_*)$ – RBF net (∞) (f) RBF kernel

FIGURE A.5: **Standard deviation $\sigma(\boldsymbol{f}_*)$ of the predictive posterior.** We perform Bayesian inference on a dataset composed by two concentric rings (SM A.4) comparing posterior uncertainties of GPs with NNGP kernels and an RBF kernel with those obtained by finite-width neural networks. Only the function space prior induced by an RBF kernel or RBF network causes epistemic uncertainties that allow outlier detection in the center or in between the two circles. However, the RBF network's uncertainties decrease with increasing input norm, which can be counteracted by further increasing σ_μ^2 (Sec. 7.3).

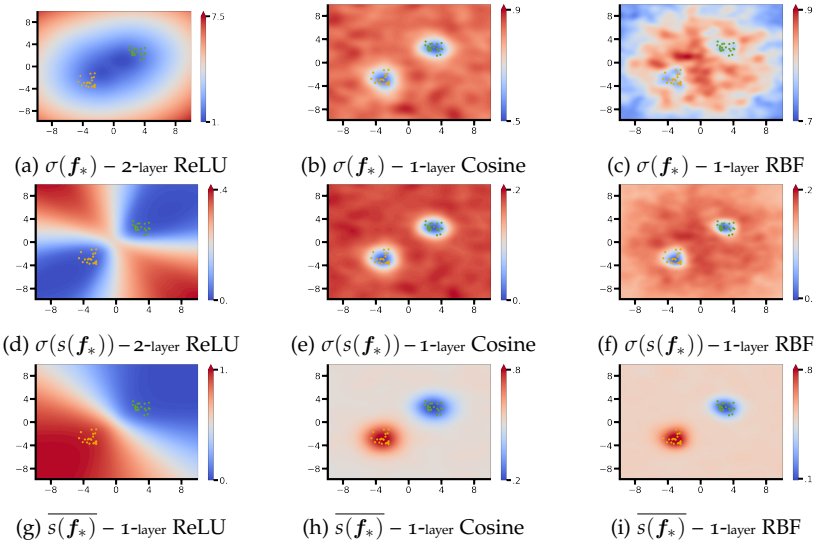
A.3.1 *2D classification*

In this section, we consider the same dataset as in Fig 7.2 in a classification setting instead of regression. In this setting, exact inference is intractable even when using GPs and approximations are needed. In particular, we study classification with the logistic likelihood function $p(\boldsymbol{y} \mid f(\boldsymbol{x}; \boldsymbol{w})) = s(-\boldsymbol{y} f(\boldsymbol{x}; \boldsymbol{w}))$ where s is the sigmoid function. We use HMC to sample from the posterior distribution in the finite-width limit, using a standard normal prior where the variance is inversely scaled by the hidden-layer's width. The results are reported in Fig A.6 for ReLU, cosine and RBF networks.

As clearly depicted in the plots, our findings regarding the importance of the prior in function space can be extended also for the classification case. Indeed, the uncertainty captured by the ReLU architecture appears unsuitable for OOD detection given that the data distribution is not captured and the disagreement is low also in regions of the 2D plane that do not contain any training data points [also see 213, 214]. This pathology appears in both cases: when the disagreement is considered over the functions $\sigma(\boldsymbol{f}_*)$ or over the sigmoid outputs $\sigma(s(\boldsymbol{f}_*))$. By contrast, also for this new choice of likelihood, the cosine and RBF architecture exhibit uncertainty that conveys similar useful properties for OOD detection as seen in the regression example in Fig. 7.2. Despite this empirical correspondence, it is important to note that the direct correspondence with the KDE technique that justifies the usefulness of OOD properties of the RBF kernel in the case of GP regression (see Sec. A.2) does not directly apply for classification. Indeed in the latter, we do not have an analytical form of the posterior as in Eq. 7.2 due to the non-Gaussianity of the logistic likelihood. Therefore our observations can not be readily generalized outside the settings of our experiments and further theoretical analysis are needed to assess whether there is an analogous justification for settings that have no closed-form solution.

A.3.2 *SplitMNIST regression*

In this section, we consider SplitMNIST tasks [103], which are binary decision tasks where the original MNIST dataset is split into five tasks (cf. Fig. 4.1). To perform exact inference via Gaussian Processes, we consider each binary decision task as regression problem with labels -1/1. For computational reasons, the training set of each task is reduced to 1000



(a) $\sigma(\boldsymbol{f}_*)$ – 2-layer ReLU (b) $\sigma(\boldsymbol{f}_*)$ – 1-layer Cosine (c) $\sigma(\boldsymbol{f}_*)$ – 1-layer RBF

(d) $\sigma(s(\boldsymbol{f}_*))$ – 2-layer ReLU (e) $\sigma(s(\boldsymbol{f}_*))$ – 1-layer Cosine (f) $\sigma(s(\boldsymbol{f}_*))$ – 1-layer RBF

(g) $\overline{s(\boldsymbol{f}_*)}$ – 1-layer ReLU (h) $\overline{s(\boldsymbol{f}_*)}$ – 1-layer Cosine (i) $\overline{s(\boldsymbol{f}_*)}$ – 1-layer RBF

FIGURE A.6: **Standard deviation over the logits $\boldsymbol{f}_*$ and mean and standard deviation over the sigmoid outputs $s(\boldsymbol{f}_*)$ for a 2D classification problem.** We perform approximate Bayesian inference with HMC for finite-width networks on a mixture of two Gaussians dataset considering different priors in function space induced by different architectural choices. The problem is now treated as classification task using HMC to sample from the posterior. We show the standard deviation $\sigma(\boldsymbol{f}_*)$ of logits $\boldsymbol{f}_*$ in **(a, b, c)**, the standard deviation $\sigma(s(\boldsymbol{f}_*))$ of the predicted probability for the input being in the positive class in **(d, e, f)**, and the corresponding mean $\overline{s(\boldsymbol{f}_*)}$ **(g, h, i)**. The ReLU network has 5 hidden units, the cosine has 100 hidden units, and the RBF network has 500 hidden units.

	Acc. Train	Acc. Test	AUROC SM	AUROC FM
RBF ($l = 1$)	$100.0^{\pm.000}$	$99.35^{\pm.577}$	$.849^{\pm.077}$	$.893^{\pm.062}$
RBF ($l = 5$)	$100.0^{\pm.000}$	$\mathbf{99.45^{\pm.552}}$	$\mathbf{.892^{\pm.057}}$	$.979^{\pm.010}$
RBF ($l = 10$)	$100.0^{\pm.000}$	$99.42^{\pm.505}$	$.884^{\pm.060}$	$.990^{\pm.005}$
ReLU 1-layer	$99.74^{\pm.313}$	$98.86^{\pm1.01}$	$.884^{\pm.060}$	$.995^{\pm.002}$
ERF 1-layer	$99.68^{\pm.356}$	$98.76^{\pm1.11}$	$.884^{\pm.060}$	$\mathbf{.996^{\pm.002}}$
Cosine 1-layer	$99.68^{\pm.356}$	$98.88^{\pm.985}$	$.880^{\pm.062}$	$.993^{\pm.003}$

TABLE A.1: **GP regression on SplitMNIST tasks.** We perform GP regression on a single SplitMNIST task using the kernels reported in the first column (entries show mean and standard deviation when using different SplitMNIST tasks for training). AUROC values are computed when considering the test data from all remaining SplitMNIST tasks as OOD data (column *SM*), or by taking the test data of FashionMNIST as OOD (column *FM*).

samples, but test sets remain at their original size. Results are reported in Table A.1.

For the chosen length-scale parameters l, RBF kernels show best generalization. Though, they don't excel at OOD detection compared to other kernel choices. In SM A.2, we discussed that epistemic uncertainties as induced by the RBF kernel can be viewed as an approximation to $p(\boldsymbol{x})$ when the dataset size goes to infinity $\mathcal{D} \to \infty$, and the length-scale goes to zero $l \to 0$. As expected, such approximation to $p(\boldsymbol{x})$ may deteriorate on high-dimensional image data, presumably as the Euclidean distance does not capture the geometry of the image manifold.

To gain a better intuition on why the RBF kernel does not excel in OOD detection for image data, we visualize the uncertainty behavior in Fig. A.7. The figure shows 2D linear subspaces of the 784D image space. These subspaces are determined by three images (see caption for details). As can be seen in Fig. A.7b and A.7e, a too small length-scale might cause test points to be not included in the low-uncertainty regions. On the other hand, if the length-scale is set too high, OOD points may fall inside low-uncertainty regions. The situations depicted in the figure show that for the given training set there is no trade-off length-scale l that prevents such behavior (because some OOD points have less Euclidean distance to training points than some test points). Using an RBF kernel with a

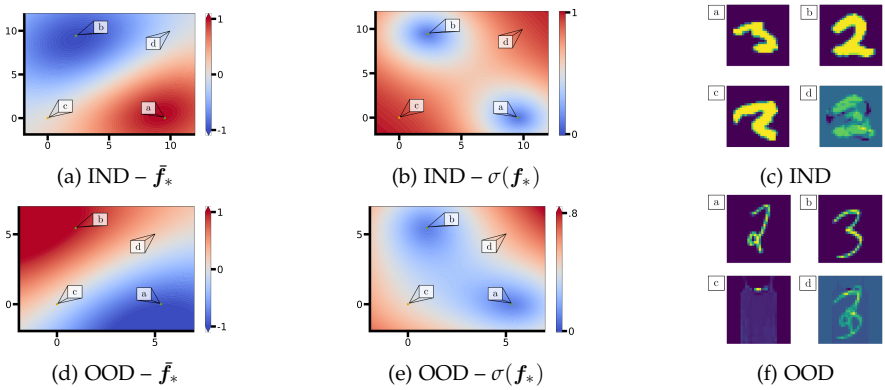
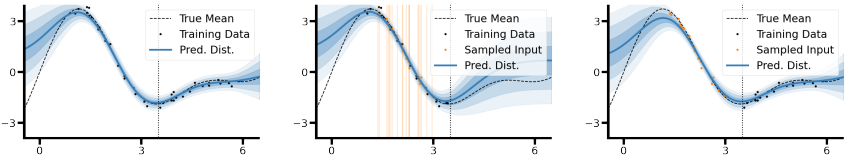


FIGURE A.7: **2D visualizations of the SplitMNIST posterior for a GP with RBF kernel** $(l = 5)$**.** The plotted 2D linear subspaces are determined by 3 images (colored dots, denoted a, b and c). In the upper row, the yellow dot represents the in-distribution (IND) test sample with the highest uncertainty (see image c in subplot **(c)**), while the two green dots are the (in-distribution) training samples with smallest Euclidean distance to image c. In the lower row, the yellow dot represents the OOD test image with lowest uncertainty (see image c in subplot **(f)**). Again, the two green dots are the closest IND training samples. Image d is in both cases a randomly chosen point on the 2D subspace. Subplots **(a)** and **(d)** show the posterior mean $\sigma(f_*)$, and subplots **(b)** and **(e)** the posterior's standard deviation $\sigma(f_*)$. Subplots **(c)** and **(f)** show the images corresponding to the 4 highlighted points on the 2D subspaces.



(a) Data from all tasks (b) Data from task 1 (c) Data from task 2 + replay

FIGURE A.8: **Continual learning with uncertainty-based replay.** As mentioned in Sec. 7.7, if uncertainty should only be low on in-distribution data, a generative model can be obtained by sampling from regions of low uncertainty. Here, we use this idea to realize a replay-based continual learning algorithm [111] without the need of storing data or maintaining a separate generative model. In this case, we split a polynomial regression dataset into two tasks (denoted by the black vertical bar). The posteriors in this figure are obtained via GP regression using an NNGP kernel corresponding to a 1-layer Cosine network. **(a)** All data is seen at once (no continual learning). **(b)** Only data of task 1 is seen to obtain the posterior. The orange vertical lines denote input locations sampled from low uncertainty regions via rejection sampling. Task 1's posterior is then used to label those input locations (orange dots). **(c)** The replay samples generated with the model of task 1 (orange dots) are used together with the training data of task 2 to obtain a combined posterior over all tasks.

metric that encodes similarities in image space rather than using Euclidean distance might overcome these problems.

A.3.3 *Continual learning via uncertainty-based replay*

In Sec. 7.7 we mention that the desideratum of having low uncertainty only on in-distribution inputs opens the possibility of considering uncertainty as an energy function from which we can sample. Thus, the uncertainty landscape can be used to construct a generative model.

In this section, we use continual learning [96] to demonstrate this conceptual idea. In continual learning, a sequence of tasks $\mathcal{D}^{(1)}, \ldots, \mathcal{D}^{(T)}$ is learned sequentially such that each task has to be learned without access to data from past or future tasks (cf. Sec. 4.1). A common approach to continual learning is via replay as introduced in Sec. 4.1.1. In this case, the current task t is learned with the available training data $\mathcal{D}^{(t)}$ as well as datasets $\tilde{\mathcal{D}}^{(1)}, \ldots, \tilde{\mathcal{D}}^{(t-1)}$ which are supposed to represent the data distributions of the previous tasks. We have seen that $\tilde{\mathcal{D}}^{(s)}$ can be constructed by either

storing data [111] or by replaying (fake) data, which ideally makes use of a generative model [e.g., 39, 115].

We consider this generative-replay scenario without the need of maintaining a separate generative model. In Fig. A.8, we consider a regression problem split into two tasks. Note, that learning a series of 1D regression problems is challenging for most continual learning algorithms, especially those relying on the recursive Bayesian update (cf. Sec. 4.1.2.1). By contrast, generative-replay methods can solve this task with ease as the structure of the data in such low-dimensional problem is easy to capture by a generative model.

In our experiment, two tasks are created by splitting the dataset of a 1D function into two parts as illustrated in Fig. A.8a. The left half is considered the first task, and the right half the second task, respectively. The blue posteriors shown in all three subpanels are obtained with a GP using the NNGP kernel of a 1-layer Cosine network, which should exhibit OOD properties similar to an RBF kernel (cf. Sec. 7.4). In Fig. A.8a, the posterior is obtained using the combined datasets from both tasks (no continual learning). In Fig. A.8b, the posterior is obtained using only the training data of the first task. After this (first task's) posterior is obtained, we can use it to perform pseudo-replay as outlined below to generate $\tilde{\mathcal{D}}^{(1)}$ which can be mixed with the data of the second task $\mathcal{D}^{(2)}$ to produce the posterior in Fig. A.8c. The posterior in Fig. A.8c looks qualitatively similar to the one in Fig. A.8a even though it has been obtained without direct access to the first task's training data $\mathcal{D}^{(1)}$.

To generate $\tilde{\mathcal{D}}^{(1)}$ while learning the second task, we need access to the posterior of the first task. Note, the model from the previous task is often kept in memory by continual learning algorithms, for instance, to use it for regularization purposes [118] or to replay data with previously trained generative models [115].[1] We generate in-distribution input locations (denoted by orange vertical bars in Fig. A.8b) via rejection sampling as in Fig. 7.7. For that, we define epistemic uncertainty as an energy function $E(\boldsymbol{x}_*) \equiv \sigma\big(\boldsymbol{f}_*(\boldsymbol{x}_*)\big)$ and construct a Boltzmann distribution $\tilde{p}(\boldsymbol{x}_*) \propto \exp\big(-E(\boldsymbol{x}_*)/T\big)$, where

1 As we use a non-parametric model (a GP) which is represented by the training data $\mathcal{D}^{(1)}$, keeping the model in memory requires to store $\mathcal{D}^{(1)}$ too, which is a violation of continual learning desiderata. However, this is just a conceptual example. If the same experiment would be performed via approximate inference in a parametric model (e.g., HMC on a corresponding finite-width network), then the (approximate) posterior of the first task can be stored without storing $\mathcal{D}^{(1)}$. In the case exact inference can be performed, the Bayesian recursive update yields a sufficient continual learning algorithm.

we set the temperature $T = 1$.[2] As uncertainty of the posterior of Fig. A.8b is only low for in-distribution inputs, we can assume obtained input locations are similar to those represented in $\mathcal{D}^{(1)}$. To construct a dataset $\tilde{\mathcal{D}}^{(1)}$, we use the posterior of Fig. A.8b to sample predictions $\boldsymbol{y}_*$ (orange dots in Fig. A.8b and Fig. A.8c).

In summary, this section presents an example application that can arise if the desideratum of high uncertainty on all OOD inputs (i.e., robust OOD detection) is fulfilled by a Bayesian model. Whether BNNs can be constructed to fulfill this desideratum on real-world data is, however, unclear at this point.

A.4 EXPERIMENTAL DETAILS

In this section, we report details on our implementation for the experiments conducted in this work. In all two-dimensional experiments: the likelihood is fixed to be Gaussian with variance 0.02. Unless noted otherwise, the prior in weight space is always a width-aware centered Gaussian with $\sigma_w^2 = 1.0$ and $\sigma_b^2 = 1.0$ except for the experiments involving RBF networks where we select $\sigma_w^2 = 200.0$. The RBF kernel bandwidth was fixed to 1.0 in all corresponding experiments.

1D REGRESSION. We construct the 1D regression example (e.g., Fig 7.1a) by defining $p(\boldsymbol{x})$ uniformly within the ranges $[1.0, 1.3]$, $[3.5, 3.8]$ and $[5.2, 5.5]$, and $p(\boldsymbol{y} \mid \boldsymbol{x})$ as $f(x) + \epsilon$ with $f(x) = 2\sin(x) + \sin(\sqrt{2}x) + \sin(\sqrt{3}x)$ and $\epsilon \sim \mathcal{N}(0, 0.2^2)$. The training set has size 20.

PERIODIC 1D REGRESSION (ONLY FIG. 7.6). We construct this 1D regression example by defining $p(\boldsymbol{x})$ uniformly within the range $[0.0, 12.5]$, and $p(\boldsymbol{y} \mid \boldsymbol{x})$ as $f(x) + \epsilon$ with $f(x) = \sin(x)$ and $\epsilon \sim \mathcal{N}(0, 3^2)$. The training set has size 30.

GAUSSIAN MIXTURE. We created a two-dimensional mixture of two Gaussians with means $\mu_1 = (-2, -2)$, $\mu_2 = (2, 2)$ and covariance $\Sigma = 0.5 \cdot \mathbb{I}$ and sampled 20 training data points.

2 Note, if $E(\boldsymbol{x}_*) \propto -\log p(\boldsymbol{x}_*)$, then this process yields exact samples from the input distribution. For instance, considering the limiting case of SM A.2 where $\sigma(\boldsymbol{f}_*(\boldsymbol{x}_*))^2 \approx 1 - p(\boldsymbol{x}_*)$, a reasonable choice of energy could be $E(\boldsymbol{x}_*) \equiv -\log\left[1 - \sigma(\boldsymbol{f}_*(\boldsymbol{x}_*))^2\right]$).

TWO RINGS. We uniformly sampled 50 training data points from two rings centred in $(0, 0)$ with inner and outer radii $R_{i,1} = 3, R_{o,1} = 4$, $R_{i,2} = 8, R_{o,2} = 9$, respectively.

HMC. We use 5 parallel chains for 5000 steps with each constituting 50 leapfrog steps with stepsize 0.001 for width 5 and 0.0001 for width 100. We considered a burn-in phase of 1000 steps and collected a total of 1000 samples.

BIBLIOGRAPHY

1. Cervera, M. R., Dätwyler, R., D'Angelo, F., Keurti, H., Grewe, B. F. & Henning, C. *Uncertainty estimation under model misspecification in neural network regression* in *NeurIPS Workshop on Robustness and misspecification in probabilistic modeling* (2021).
2. Jegminat, J., Jastrzębowska, M. A., Pachai, M. V., Herzog, M. H. & Pfister, J.-P. Bayesian regression explains how human participants handle parameter uncertainty. *PLoS Computational Biology* **16** (2020).
3. Cox, R. T. Probability, frequency and reasonable expectation. *American journal of physics* **14**, 1 (1946).
4. Jaynes, E. T. *Probability theory: The logic of science* (Cambridge university press, 2003).
5. Bowers, J. S. & Davis, C. J. Bayesian just-so stories in psychology and neuroscience. *Psychological Bulletin* **138**, 389 (2012).
6. Marcus, G. F. & Davis, E. How Robust Are Probabilistic Models of Higher-Level Cognition? *Psychological Science* **24**, 2351 (2013).
7. Goodman, N. D., Frank, M. C., Griffiths, T. L., Tenenbaum, J. B., Battaglia, P. W. & Hamrick, J. B. Relevant and Robust: A Response to Marcus and Davis (2013). *Psychological Science* **26**, 539 (2015).
8. Lakshminarayanan, B., Pritzel, A. & Blundell, C. *Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles* in *Advances in Neural Information Processing Systems* (eds Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S. & Garnett, R.) **30** (Curran Associates, Inc., 2017).
9. D'Angelo, F. & Fortuin, V. *Repulsive Deep Ensembles are Bayesian* in *Thirty-Fifth Conference on Neural Information Processing Systems* (2021).
10. Aitchison, L., Jegminat, J., Menendez, J. A., Pfister, J.-P., Pouget, A. & Latham, P. E. Synaptic plasticity as Bayesian inference. *Nature Neuroscience* **24**, 565 (2021).
11. Sandberg, A., Drexler, E. & Ord, T. Dissolving the Fermi paradox. *arXiv* (2018).
12. Hensch, T. K. Critical period regulation. *Annual Review of Neuroscience* **27**, 549 (2004).

13. McCloskey, M. & Cohen, N. Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem. *Psychology of Learning and Motivation - Advances in Research and Theory* **24**, 109 (1989).

14. Ratcliff, R. Connectionist models of recognition memory: Constraints imposed by learning and forgetting functions. *Psychological Review* **97**, 285 (1990).

15. Marr, D. & Brindley, G. S. Simple memory: a theory for archicortex. *Philosophical Transactions of the Royal Society of London. B, Biological Sciences* **262**, 23 (1971).

16. McClelland, J. L., McNaughton, B. L. & O'Reilly, R. C. Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory. *Psychological review* **102**, 419 (1995).

17. Hassabis, D., Kumaran, D., Summerfield, C. & Botvinick, M. Neuroscience-Inspired Artificial Intelligence. *Neuron* **95**, 245 (2017).

18. Hadsell, R., Rao, D., Rusu, A. A. & Pascanu, R. Embracing Change: Continual Learning in Deep Neural Networks. *Trends in Cognitive Sciences* **24**, 1028 (2020).

19. Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L. D., Monfort, M., Muller, U., Zhang, J., *et al.* End to end learning for self-driving cars. *arXiv* (2016).

20. Hastie, T., Tibshirani, R. & Friedman, J. in *The Elements of Statistical Learning: Data Mining, Inference, and Prediction* (Springer, New York, NY, 2009).

21. Mohri, M., Rostamizadeh, A. & Talwalkar, A. *Foundations of Machine Learning* (The MIT Press, 2018).

22. Sugiyama, M. & Kawanabe, M. *Machine Learning in Non-Stationary Environments: Introduction to Covariate Shift Adaptation* (The MIT Press, 2012).

23. Cybenko, G. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems* **2**, 303 (1989).

24. Hornik, K. Approximation capabilities of multilayer feedforward networks. *Neural Networks* **4**, 251 (1991).

25. Owhadi, H., Scovel, C. & Sullivan, T. On the Brittleness of Bayesian Inference. *SIAM Review* **57**. Publisher: Society for Industrial and Applied Mathematics, 566 (2015).

26. Goodfellow, I., Bengio, Y. & Courville, A. *Deep Learning* (MIT Press, 2016).
27. McCulloch, W. S. & Pitts, W. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics* **5**, 115 (1943).
28. Rosenblatt, F. *The Perceptron, a Perceiving and Recognizing Automaton Project Para* (Cornell Aeronautical Laboratory, 1957).
29. Abbott, L. F. Lapicque's introduction of the integrate-and-fire model neuron (1907). *Brain Research Bulletin* **50**, 303 (1999).
30. LeCun, Y., Bottou, L., Bengio, Y. & Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* **86**, 2278 (1998).
31. Elman, J. L. Finding structure in time. *Cognitive Science* **14**, 179 (1990).
32. Hochreiter, S. & Schmidhuber, J. Long short-term memory. *Neural computation* **9**, 1735 (1997).
33. Ehret, B., Henning, C., Cervera, M. R., Meulemans, A., von Oswald, J. & Grewe, B. F. *Continual Learning in Recurrent Neural Networks* in *International Conference on Learning Representations* (2021).
34. Rumelhart, D. E., Hinton, G. E. & Williams, R. J. Learning representations by back-propagating errors. *Nature* **323**, 533 (1986).
35. Ha, D., Dai, A. & Le, Q. *HyperNetworks* in *International Conference on Learning Representations* (2017).
36. Schmidhuber, J. Learning to Control Fast-Weight Memories: An Alternative to Dynamic Recurrent Networks. *Neural Comput.* **4**, 131 (1992).
37. Rusu, A. A., Rao, D., Sygnowski, J., Vinyals, O., Pascanu, R., Osindero, S. & Hadsell, R. *Meta-Learning with Latent Embedding Optimization* in *International Conference on Learning Representations* (2019).
38. He, X., Sygnowski, J., Galashov, A., Rusu, A. A., Teh, Y. W. & Pascanu, R. Task agnostic continual learning via meta learning. *arXiv* (2019).
39. Von Oswald, J., Henning, C., Sacramento, J. & Grewe, B. F. *Continual learning with hypernetworks* in *International Conference on Learning Representations* (2020).
40. Henning, C., Cervera, M. R., D'Angelo, F., von Oswald, J., Traber, R., Ehret, B., Kobayashi, S., Grewe, B. F. & Sacramento, J. *Posterior Meta-Replay for Continual Learning* in *Conference on Neural Information Processing Systems* (2021).

41. Louizos, C. & Welling, M. *Multiplicative Normalizing Flows for Variational Bayesian Neural Networks* in *Proceedings of the 34th International Conference on Machine Learning - Volume 70* (JMLR.org, 2017), 2218.

42. Pawlowski, N., Brock, A., Lee, M. C., Rajchl, M. & Glocker, B. Implicit weight uncertainty in neural networks. *arXiv* (2017).

43. Krueger, D., Huang, C.-W., Islam, R., Turner, R., Lacoste, A. & Courville, A. Bayesian Hypernetworks. *arXiv* (2017).

44. Henning, C., von Oswald, J., Sacramento, J., Surace, S. C., Pfister, J.-P. & Grewe, B. F. *Approximating the predictive distribution via adversarially-trained hypernetworks* in *NeurIPS Workshop on Bayesian Deep Learning* (2018).

45. Gal, Y. *Uncertainty in Deep Learning* PhD thesis (University of Cambridge, 2016).

46. Hüllermeier, E. & Waegeman, W. Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods. *Machine Learning* **110**, 457 (2021).

47. Gneiting, T. & Raftery, A. E. Strictly proper scoring rules, prediction, and estimation. *Journal of the American statistical Association* **102**, 359 (2007).

48. Guo, C., Pleiss, G., Sun, Y. & Weinberger, K. Q. *On calibration of modern neural networks* in *Proceedings of the 34th International Conference on Machine Learning* (Sydney, NSW, Australia, 2017), 1321.

49. Snoek, J., Ovadia, Y., Fertig, E., Lakshminarayanan, B., Nowozin, S., Sculley, D., Dillon, J., Ren, J. & Nado, Z. *Can you trust your model's uncertainty? Evaluating predictive uncertainty under dataset shift* in *Advances in Neural Information Processing Systems* (2019), 13969.

50. Efron, B. Bayesians, frequentists, and scientists. *Journal of the American Statistical Association* **100**, 1 (2005).

51. Ghahramani, Z. Probabilistic machine learning and artificial intelligence. *Nature* **521** (2015).

52. Levin, E., Tishby, N. & Solla, S. A statistical approach to learning and generalization in layered neural networks. *Proceedings of the IEEE* **78**, 1568 (1990).

53. MacKay, D. J. *Information theory, inference and learning algorithms* (Cambridge university press, 2003).

54. Bruinsma, W., Foong, A. Y. K. & Turner, R. E. *What Keeps a Bayesian Awake At Night? Part 1: Day Time* `https://mlg-blog.com/2021/03/31/what-keeps-a-bayesian-awake-at-night-part-1.html` (2021).

55. Efron, B. Why Isn't Everyone a Bayesian? *The American Statistician* **40**, 1 (1986).

56. Murray, I. & Ghahramani, Z. *A note on the evidence and Bayesian Occam's razor* tech. rep. GCNU-TR 2005-003 (Gatsby Computational Neuroscience Unit, University College London, 2005).

57. MacKay, D. J. C. in *Maximum Entropy and Bayesian Methods: Seattle, 1991* (eds Smith, C. R., Erickson, G. J. & Neudorfer, P. O.) 39 (Springer Netherlands, Dordrecht, 1992).

58. Diaconis, P. & Freedman, D. On the consistency of Bayes estimates. *The Annals of Statistics*, 1 (1986).

59. Van der Vaart, A. W. *Asymptotic statistics* (Cambridge university press, 2000).

60. Kleijn, B. J. & van der Vaart, A. W. The Bernstein-von-Mises theorem under misspecification. *Electronic Journal of Statistics* **6**, 354 (2012).

61. Rasmussen, C. E. in *Advanced Lectures on Machine Learning: ML Summer Schools 2003, Canberra, Australia, February 2 - 14, 2003, Tübingen, Germany, August 4 - 16, 2003, Revised Lectures* 63 (Springer Berlin Heidelberg, Berlin, Heidelberg, 2004).

62. Neal, R. M. *Bayesian Learning for Neural Networks* (Springer-Verlag, Berlin, Heidelberg, 1996).

63. Keeling, C. D. & Whorf, T. P. Monthly Atmospheric CO2 Records from Sites in the SIO Air Sampling Network (2004).

64. Gelman, A. Objections to Bayesian statistics. *Bayesian Analysis* **3**, 445 (2008).

65. Fortuin, V. Priors in Bayesian deep learning: A review. *arXiv* (2021).

66. Matsubara, T., Oates, C. J. & Briol, F.-X. The ridgelet prior: A covariance function approach to prior specification for bayesian neural networks. *Journal of Machine Learning Research* **22**, 1 (2021).

67. Wilson, A. G. & Izmailov, P. *Bayesian deep learning and a probabilistic perspective of generalization* in *Advances in Neural Information Processing Systems* (2020).

68. Quiñonero-Candela, J., Sugiyama, M., Lawrence, N. D. & Schwaighofer, A. *Dataset shift in machine learning* (Mit Press, 2009).

69. Wolpert, D. H. The Existence of A Priori Distinctions Between Learning Algorithms. *Neural Computation* **8**, 1391 (1996).

70. Schaffer, C. in *Machine Learning Proceedings 1994* (eds Cohen, W. W. & Hirsh, H.) 259 (Morgan Kaufmann, San Francisco (CA), 1994).

71. Rao, R. B., Gordon, D. & Spears, W. *For Every Generalization Action, Is There Really An Equal And Opposite Reaction? Analysis of the Conservation Law for Generalization Performance* in *Proceedings of the 12th International Conference on Machine Learning* (Morgan Kaufmann, 1995), 471.

72. Tishby, Levin & Solla. *Consistent inference of probabilities in layered networks: predictions and generalizations* in *International 1989 Joint Conference on Neural Networks* **2** (1989), 403.

73. MacKay, D. J. A practical Bayesian framework for backpropagation networks. *Neural computation* **4**, 448 (1992).

74. Murphy, K. P. *Machine Learning: A Probabilistic Perspective* (ed Bach, F.) (MIT Press, Cambridge, MA, USA, 2012).

75. Blei, D. M., Kucukelbir, A. & McAuliffe, J. D. Variational Inference: A Review for Statisticians. *Journal of the American Statistical Association* **112**, 859 (2017).

76. Graves, A. *Practical Variational Inference for Neural Networks* in *Advances in Neural Information Processing Systems* (eds Shawe-Taylor, J., Zemel, R., Bartlett, P., Pereira, F. & Weinberger, K. Q.) **24** (Curran Associates, Inc., 2011).

77. Blundell, C., Cornebise, J., Kavukcuoglu, K. & Wierstra, D. *Weight Uncertainty in Neural Networks* in *Proceedings of the 32nd International Conference on Machine Learning* (eds Bach, F. & Blei, D.) **37** (PMLR, Lille, France, 2015), 1613.

78. Ambrogioni, L., Güçlü, U., Güçlütürk, Y., Hinne, M., van Gerven, M. A. J. & Maris, E. *Wasserstein Variational Inference* in *Advances in Neural Information Processing Systems* (eds Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N. & Garnett, R.) **31** (Curran Associates, Inc., 2018).

79. Kingma, D. P. & Welling, M. *Auto-Encoding Variational Bayes* in *International Conference on Learning Representations* (2014).

80. Ranganath, R., Gerrish, S. & Blei, D. *Black box variational inference* in *Artificial intelligence and statistics* (2014), 814.

81. Mescheder, L., Nowozin, S. & Geiger, A. *Adversarial Variational Bayes: Unifying Variational Autoencoders and Generative Adversarial Networks* in *Proceedings of the 34th International Conference on Machine Learning* **70** (PMLR, International Convention Centre, Sydney, Australia, 2017), 2391.

82. Shi, J., Sun, S. & Zhu, J. *A Spectral Approach to Gradient Estimation for Implicit Distributions* in *Proceedings of the 35th International Conference on Machine Learning* (eds Dy, J. & Krause, A.) **80** (PMLR, Stockholmsmässan, Stockholm Sweden, 2018), 4644.

83. Kunstner, F., Hennig, P. & Balles, L. *Limitations of the empirical Fisher approximation for natural gradient descent* in *Advances in Neural Information Processing Systems* (eds Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. & Garnett, R.) **32** (Curran Associates, Inc., 2019).

84. Huszár, F. Note on the quadratic penalties in elastic weight consolidation. *Proceedings of the National Academy of Sciences* **115**, E2496 (2018).

85. Ritter, H., Botev, A. & Barber, D. *Online Structured Laplace Approximations for Overcoming Catastrophic Forgetting* in *Advances in Neural Information Processing Systems* (eds Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N. & Garnett, R.) **31** (Curran Associates, Inc., 2018), 3738.

86. Speagle, J. S. A conceptual introduction to Markov chain Monte Carlo methods. *arXiv* (2019).

87. Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H. & Teller, E. Equation of State Calculations by Fast Computing Machines. *The Journal of Chemical Physics* **21**, 1087 (1953).

88. Hastings, W. K. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika* **57**, 97 (1970).

89. Hohendorff, J. M. & Rosenthal, J. An introduction to Markov chain Monte Carlo. *University of Toronto, Department of Statistics, supervised reading report* (2005).

90. Duane, S., Kennedy, A. D., Pendleton, B. J. & Roweth, D. Hybrid monte carlo. *Physics letters B* **195**, 216 (1987).

91. Neal, R. M. *et al.* MCMC using Hamiltonian dynamics. *Handbook of Markov chain Monte Marlo* **2**, 2 (2011).

92. Izmailov, P., Vikram, S., Hoffman, M. D. & Wilson, A. G. *What Are Bayesian Neural Network Posteriors Really Like?* in *Proceedings of the 38th International Conference on Machine Learning* (2021).

93. Welling, M. & Teh, Y. W. *Bayesian learning via stochastic gradient Langevin dynamics* in *Proceedings of the 28th International Conference on Machine Learning* (2011), 681.

94. Chen, T., Fox, E. & Guestrin, C. *Stochastic gradient Hamiltonian Monte Carlo* in *Proceedings of the 31st International Conference on Machine Learning* (2014), 1683.

95. Henning, C., D'Angelo, F. & Grewe, B. F. *Are Bayesian neural networks intrinsically good at out-of-distribution detection?* in *ICML Workshop on Uncertainty and Robustness in Deep Learning* (2021).

96. Parisi, G. I., Kemker, R., Part, J. L., Kanan, C. & Wermter, S. Continual lifelong learning with neural networks: A review. *Neural Networks* **113**, 54 (2019).

97. Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., *et al.* Language models are few-shot learners. *arXiv* (2020).

98. Bender, E. M., Gebru, T., McMillan-Major, A. & Shmitchell, S. *On the Dangers of Stochastic Parrots: Can Language Models Be Too Big?* in (Association for Computing Machinery, Virtual Event, Canada, 2021), 610.

99. Prabhu, A., Torr, P. H. & Dokania, P. K. *GDumb: A simple approach that questions our progress in continual learning* in *European Conference on Computer Vision* (2020), 524.

100. Liu, G.-H. & Theodorou, E. A. Deep learning theory review: An optimal control and dynamical systems perspective. *arXiv* (2019).

101. Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H. & He, Q. A comprehensive survey on transfer learning. *Proceedings of the IEEE* **109**, 43 (2020).

102. Schwarz, J., Czarnecki, W., Luketina, J., Grabska-Barwinska, A., Teh, Y. W., Pascanu, R. & Hadsell, R. *Progress & Compress: A scalable framework for continual learning* in *Proceedings of the 35th International Conference on Machine Learning* (eds Dy, J. & Krause, A.) **80** (PMLR, Stockholmsmässan, Stockholm Sweden, 2018), 4528.

103. Zenke, F., Poole, B. & Ganguli, S. *Continual Learning Through Synaptic Intelligence* in *Proceedings of the 34th International Conference on Machine Learning - Volume 70* (JMLR.org, 2017), 3987.

104. Van de Ven, G. M. & Tolias, A. S. Generative replay with feedback connections as a general strategy for continual learning. *arXiv* (2018).

105. Van de Ven, G. M. & Tolias, A. S. Three scenarios for continual learning. *arXiv* (2019).

106. Van de Ven, G. M., Siegelmann, H. T. & Tolias, A. S. Brain-inspired replay for continual learning with artificial neural networks. *Nature communications* **11**, 1 (2020).

107. Yu, L., Twardowski, B., Liu, X., Herranz, L., Wang, K., Cheng, Y., Jui, S. & Weijer, J. v. d. *Semantic drift compensation for class-incremental learning* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020), 6982.

108. Li, S., Du, Y., van de Ven, G. M., Torralba, A. & Mordatch, I. Energy-Based Models for Continual Learning. *arXiv* (2020).

109. Li, A., Boyd, A. J., Smyth, P. & Mandt, S. *Variational Beam Search for Novelty Detection* in *Third Symposium on Advances in Approximate Bayesian Inference* (2021).

110. Kurle, R., Cseke, B., Klushyn, A., van der Smagt, P. & Günnemann, S. *Continual Learning with Bayesian Neural Networks for Non-Stationary Data* in *International Conference on Learning Representations* (2020).

111. Lin, L.-J. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning* **8**, 293 (1992).

112. Rebuffi, S.-A., Kolesnikov, A., Sperl, G. & Lampert, C. H. *icarl: Incremental classifier and representation learning* in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition* (2017), 2001.

113. Borsos, Z., Mutny, M. & Krause, A. *Coresets via Bilevel Optimization for Continual Learning and Streaming* in *Advances in Neural Information Processing Systems* (eds Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F. & Lin, H.) **33** (Curran Associates, Inc., 2020), 14879.

114. Hinton, G., Vinyals, O. & Dean, J. Distilling the knowledge in a neural network. *arXiv* (2015).

115. Shin, H., Lee, J. K., Kim, J. & Kim, J. in *Advances in Neural Information Processing Systems 30* (eds Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S. & Garnett, R.) 2990 (Curran Associates, Inc., 2017).

116. D'Angelo, F. & Henning, C. *Uncertainty-based out-of-distribution detection requires suitable function space priors*

117. Farquhar, S. & Gal, Y. A Unifying Bayesian View of Continual Learning. *Bayesian Deep Learning Workshop at NeurIPS* (2018).

118. Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D. & Hadsell, R. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences* **114**, 3521 (2017).

119. Nguyen, C. V., Li, Y., Bui, T. D. & Turner, R. E. *Variational Continual Learning* in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings* (2018).

120. Swaroop, S., Nguyen, C. V., Bui, T. D. & Turner, R. E. Improving and Understanding Variational Continual Learning. *Continual Learning Workshop at NeurIPS* (2018).

121. Loo, N., Swaroop, S. & Turner, R. E. *Generalized Variational Continual Learning* in *International Conference on Learning Representations* (2021).

122. Rusu, A. A., Rabinowitz, N. C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R. & Hadsell, R. Progressive neural networks. *arXiv* (2016).

123. Cossu, A., Carta, A. & Bacciu, D. *Continual Learning with Gated Incremental Memories for sequential data processing* in *2020 International Joint Conference on Neural Networks (IJCNN)* (2020), 1.

124. Van de Ven, G. M., Li, Z. & Tolias, A. S. *Class-Incremental Learning With Generative Classifiers* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops* (2021), 3611.

125. Lee, S., Ha, J., Zhang, D. & Kim, G. *A Neural Dirichlet Process Mixture Model for Task-Free Continual Learning* in *International Conference on Learning Representations* (2020).

126. Nalisnick, E., Matsukawa, A., Teh, Y. W., Gorur, D. & Lakshminarayanan, B. *Do Deep Generative Models Know What They Don't Know?* in *International Conference on Learning Representations* (2019).

127. Masse, N. Y., Grant, G. D. & Freedman, D. J. Alleviating catastrophic forgetting using context-dependent gating and synaptic stabilization. *Proceedings of the National Academy of Sciences* **115**, E10467 (2018).

128. Aljundi, R., Babiloni, F., Elhoseiny, M., Rohrbach, M. & Tuytelaars, T. *Memory aware synapses: Learning what (not) to forget* in *Proceedings of the European Conference on Computer Vision (ECCV)* (2018), 139.

129. Benzing, F. Unifying Regularisation Methods for Continual Learning. *arXiv* (2020).

130. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S. & Hassabis, D. Human-level control through deep reinforcement learning. *Nature* **518**, 529 (2015).

131. Zeno, C., Golan, I., Hoffer, E. & Soudry, D. Task-Agnostic Continual Learning Using Online Variational Bayes With Fixed-Point Updates. *Neural Computation* **33**, 3139 (2021).

132. Liu, G.-H. & Theodorou, E. A. Deep learning theory review: An optimal control and dynamical systems perspective. *arXiv* (2019).

133. Hospedales, T., Antoniou, A., Micaelli, P. & Storkey, A. Meta-learning in neural networks: A survey. *arXiv* (2020).

134. Beaulieu, S., Frati, L., Miconi, T., Lehman, J., Stanley, K. O., Clune, J. & Cheney, N. Learning to Continually Learn. *Santiago de Compostela*, 10 (2020).

135. Robins, A. Catastrophic Forgetting, Rehearsal and Pseudorehearsal. *Connection Science* **7**, 123 (1995).

136. He, X. & Jaeger, H. *Overcoming Catastrophic Interference using Conceptor-Aided Backpropagation* in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings* (2018).

137. Wu, C., Herranz, L., Liu, X., wang yaxing, y., van de Weijer, J. & Raducanu, B. in *Advances in Neural Information Processing Systems 31* (eds Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N. & Garnett, R.) 5962 (Curran Associates, Inc., 2018).

138. Bertinetto, L., Henriques, J. F., Valmadre, J., Torr, P. & Vedaldi, A. *Learning feed-forward one-shot learners* in *Advances in Neural Information Processing Systems 29* (eds Lee, D. D., Sugiyama, M., Luxburg, U. V., Guyon, I. & Garnett, R.) (Curran Associates, Inc., 2016), 523.

139. Jia, X., De Brabandere, B., Tuytelaars, T. & Gool, L. V. in *Advances in Neural Information Processing Systems 29* (eds Lee, D. D., Sugiyama, M., Luxburg, U. V., Guyon, I. & Garnett, R.) 667 (Curran Associates, Inc., 2016).

140. Li, Z. & Hoiem, D. Learning without Forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **40**, 2935 (2018).

141. Benjamin, A., Rolnick, D. & Kording, K. *Measuring and regularizing networks in function space* in *International Conference on Learning Representations* (2019).

142. Kingma, D. P. & Ba, J. *Adam: A Method for Stochastic Optimization* in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings* (2015).

143. Farquhar, S. & Gal, Y. Towards Robust Evaluations of Continual Learning. *Lifelong Learning: A Reinforcement Learning Approach Workshop at ICML* (2018).

144. Hendrycks, D. & Gimpel, K. *A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks* in *International Conference on Learning Representations* (2017).

145. Liang, S., Li, Y. & Srikant, R. *Enhancing The Reliability of Out-of-distribution Image Detection in Neural Networks* in *International Conference on Learning Representations* (2018).

146. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. & Bengio, Y. in *Advances in Neural Information Processing Systems 27* (eds Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D. & Weinberger, K. Q.) 2672 (Curran Associates, Inc., 2014).

147. He, K., Zhang, X., Ren, S. & Sun, J. *Deep residual learning for image recognition* in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), 770.

148. Goodfellow, I. J., Mirza, M., Xiao, D., Courville, A. & Bengio, Y. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv* (2013).

149. Fernando, C., Banarse, D., Blundell, C., Zwols, Y., Ha, D., Rusu, A. A., Pritzel, A. & Wierstra, D. Pathnet: Evolution channels gradient descent in super neural networks. *arXiv preprint arXiv:1701.08734* (2017).

150. Serra, J., Suris, D., Miron, M. & Karatzoglou, A. *Overcoming Catastrophic Forgetting with Hard Attention to the Task* in *Proceedings of the 35th International Conference on Machine Learning* (eds Dy, J. & Krause, A.) **80** (PMLR, Stockholmsmässan, Stockholm Sweden, 2018), 4548.

151. Mallya, A. & Lazebnik, S. *Packnet: Adding multiple tasks to a single network by iterative pruning* in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018), 7765.

152. Lampinen, A. K. & McClelland, J. L. Embedded Meta-Learning: Toward more flexible deep-learning models. *arXiv* (2019).

153. Hu, W., Lin, Z., Liu, B., Tao, C., Tao, Z., Ma, J., Zhao, D. & Yan, R. *Overcoming Catastrophic Forgetting for Continual Learning via Model Adaptation* in *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019* (2019).

154. Mandivarapu, J. K., Camp, B. & Estrada, R. Self-Net: Lifelong Learning via Continual Self-Modeling. *Frontiers in Artificial Intelligence* **3**, 19 (2020).

155. Brea, J., Urbanczik, R. & Senn, W. A Normative Theory of Forgetting: Lessons from the Fruit Fly. *PLOS Computational Biology* **10**, e1003640 (2014).

156. Richards, B. A. & Frankland, P. W. The Persistence and Transience of Memory. *Neuron* **94**, 1071 (2017).

157. French, R. M. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences* **3**, 128 (1999).

158. Kumaran, D., Hassabis, D. & McClelland, J. L. What Learning Systems do Intelligent Agents Need? Complementary Learning Systems Theory Updated. *Trends in Cognitive Sciences* **20**, 512 (2016).

159. Hassabis, D., Kumaran, D., Summerfield, C. & Botvinick, M. Neuroscience-Inspired Artificial Intelligence. *Neuron* **95**, 245 (2017).

160. Marder, E. Neuromodulation of Neuronal Circuits: Back to the Future. *Neuron* **76**, 1 (2012).

161. Mante, V., Sussillo, D., Shenoy, K. V. & Newsome, W. T. Context-dependent computation by recurrent dynamics in prefrontal cortex. *Nature* **503**, 78 (2013).

162. Jaeger, H. Controlling Recurrent Neural Networks by Conceptors. *arXiv* (2014).

163. Stroud, J. P., Porter, M. A., Hennequin, G. & Vogels, T. P. Motor primitives in space and time via targeted gain modulation in cortical networks. *Nature Neuroscience* **21**, 1774 (2018).

164. Wortsman, M., Ramanujan, V., Liu, R., Kembhavi, A., Rastegari, M., Yosinski, J. & Farhadi, A. *Supermasks in Superposition* in *Advances in Neural Information Processing Systems* (eds Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F. & Lin, H.) **33** (Curran Associates, Inc., 2020), 15173.

165. Lee, S.-W., Kim, J.-H., Jun, J., Ha, J.-W. & Zhang, B.-T. *Overcoming Catastrophic Forgetting by Incremental Moment Matching* in *Advances in Neural Information Processing Systems* (eds Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S. & Garnett, R.) **30** (Curran Associates, Inc., 2017), 4652.

166. Li, H., Barnaghi, P., Enshaeifar, S. & Ganz, F. Continual learning using Bayesian neural networks. *IEEE Transactions on Neural Networks and Learning Systems* (2020).

167. Tuor, T., Wang, S. & Leung, K. Continual Learning Without Knowing Task Identities: Do Simple Models Work? *OpenReview* (2021).

168. K J, J. & N Balasubramanian, V. *Meta-Consolidation for Continual Learning* in *Advances in Neural Information Processing Systems* (eds Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F. & Lin, H.) **33** (Curran Associates, Inc., 2020), 14374.

169. Zhang, C., Bengio, S., Hardt, M., Recht, B. & Vinyals, O. *Understanding deep learning requires rethinking generalization* in *International Conference on Learning Representations* (2017).

170. Huszár, F. Variational inference using implicit distributions. *arXiv* (2017).

171. Papamakarios, G., Nalisnick, E., Rezende, D. J., Mohamed, S. & Lakshminarayanan, B. Normalizing flows for probabilistic modeling and inference. *arXiv* (2019).

172. Krizhevsky, A. & Hinton, G. Learning multiple layers of features from tiny images. *Master's thesis, Department of Computer Science, University of Toronto* (2009).

173. Mukhoti, J., Kirsch, A., van Amersfoort, J., Torr, P. H. & Gal, Y. Deterministic Neural Networks with Appropriate Inductive Biases Capture Epistemic and Aleatoric Uncertainty. *arXiv* (2021).

174. Lakshminarayanan, B., Tran, D., Liu, J., Padhy, S., Bedrax-Weiss, T. & Lin, Z. *Simple and Principled Uncertainty Estimation with Deterministic Deep Learning via Distance Awareness* in *Advances in Neural Information Processing Systems 33* (2020).

175. Williams, C. K. Computing with infinite networks. *Advances in neural information processing systems*, 295 (1997).

176. O'Hagan, A. Bayesian statistics: principles and benefits. *Frontis*, 31 (2004).

177. Hornik, K. Approximation capabilities of multilayer feedforward networks. *Neural Networks* **4**, 251 (1991).

178. Maddox, W. J., Izmailov, P., Garipov, T., Vetrov, D. P. & Wilson, A. G. *A Simple Baseline for Bayesian Uncertainty in Deep Learning* in *Advances in Neural Information Processing Systems* (eds Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. & Garnett, R.) **32** (Curran Associates, Inc., 2019).

179. Ciosek, K., Fortuin, V., Tomioka, R., Hofmann, K. & Turner, R. *Conservative Uncertainty Estimation By Fitting Prior Networks* in *International Conference on Learning Representations* (2020).

180. Barnett, V. & Lewis, T. Outliers in statistical data. *Wiley Series in Probability and Mathematical Statistics. Applied Probability and Statistics* (1984).

181. Zimek, A. & Filzmoser, P. There and back again: Outlier detection between statistical reasoning and data mining algorithms. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* **8**, e1280 (2018).

182. Pimentel, M. A., Clifton, D. A., Clifton, L. & Tarassenko, L. A review of novelty detection. *Signal Processing* **99**, 215 (2014).

183. Hendrycks, D., Mazeika, M. & Dietterich, T. *Deep Anomaly Detection with Outlier Exposure* in *International Conference on Learning Representations* (2019).

184. Lee, J., Sohl-dickstein, J., Pennington, J., Novak, R., Schoenholz, S. & Bahri, Y. *Deep Neural Networks as Gaussian Processes* in *International Conference on Learning Representations* (2018).

185. De G. Matthews, A. G., Hron, J., Rowland, M., Turner, R. E. & Ghahramani, Z. *Gaussian Process Behaviour in Wide Deep Neural Networks* in *International Conference on Learning Representations* (2018).

186. Cho, Y. & Saul, L. *Kernel Methods for Deep Learning* in *Advances in Neural Information Processing Systems* (eds Bengio, Y., Schuurmans, D., Lafferty, J., Williams, C. & Culotta, A.) **22** (Curran Associates, Inc., 2009).

187. Tsuchida, R., Roosta, F. & Gallagher, M. *Invariance of weight distributions in rectified MLPs* in *International Conference on Machine Learning* (2018), 4995.

188. Pang, G., Yang, L. & Karniadakis, G. E. Neural-net-induced Gaussian process regression for function approximation and PDE solution. *Journal of Computational Physics* **384**, 270 (2019).

189. Pearce, T., Tsuchida, R., Zaki, M., Brintrup, A. & Neely, A. *Expressive priors in bayesian neural networks: Kernel combinations and periodic functions* in *Uncertainty in Artificial Intelligence* (2020), 134.

190. Broomhead, D. S. & Lowe, D. *Radial basis functions, multi-variable functional interpolation and adaptive networks* tech. rep. (Royal Signals and Radar Establishment Malvern (United Kingdom), 1988).

191. Valiant, L. G. A theory of the learnable. *Communications of the ACM* **27**, 1134 (1984).

192. McAllester, D. A. Some PAC-Bayesian Theorems. *Machine Learning* **37**, 355 (1999).

193. Catoni, O. Pac-Bayesian Supervised Classification: The Thermodynamics of Statistical Learning. *Lecture Notes-Monograph Series* **56**, i (2007).

194. Seeger, M. Pac-Bayesian Generalisation Error Bounds for Gaussian Process Classification. *J. Mach. Learn. Res.* **3**, 233 (2003).

195. Reeb, D., Doerr, A., Gerwinn, S. & Rakitsch, B. *Learning Gaussian Processes by Minimizing PAC-Bayesian Generalization Bounds* in *Advances in Neural Information Processing Systems* (eds Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N. & Garnett, R.) **31** (Curran Associates, Inc., 2018).

196. Germain, P., Bach, F., Lacoste, A. & Lacoste-Julien, S. *PAC-Bayesian Theory Meets Bayesian Inference* in *Advances in Neural Information Processing Systems* (eds Lee, D., Sugiyama, M., Luxburg, U., Guyon, I. & Garnett, R.) **29** (Curran Associates, Inc., 2016).

197. Burt, D. R., Ober, S. W., Garriga-Alonso, A. & van der Wilk, M. *Understanding Variational Inference in Function-Space* in *Third Symposium on Advances in Approximate Bayesian Inference* (2021).

198. Zięba, M., Przewięźlikowski, M., Śmieja, M., Tabor, J., Trzcinski, T. & Spurek, P. RegFlow: Probabilistic Flow-based Regression for Future Prediction. *arXiv* (2020).

199. Flam-Shepherd, D., Requeima, J. & Duvenaud, D. *Mapping Gaussian process priors to Bayesian neural networks* in *NIPS Bayesian deep learning workshop* (2017).

200. Sun, S., Zhang, G., Shi, J. & Grosse, R. *Functional Variational Bayesian Neural Networks* in *International Conference on Learning Representations* (2019).

201. Tran, B.-H., Rossi, S., Milios, D. & Filippone, M. All you need is a good functional prior for bayesian deep learning. *arXiv* (2020).

202. Izmailov, P., Nicholson, P., Lotfi, S. & Wilson, A. G. Dangers of Bayesian Model Averaging under Covariate Shift. *arXiv* (2021).

203. Liu, Q., Lee, J. & Jordan, M. *A Kernelized Stein Discrepancy for Goodness-of-fit Tests* in *Proceedings of The 33rd International Conference on Machine Learning* (eds Balcan, M. F. & Weinberger, K. Q.) **48** (PMLR, New York, New York, USA, 2016), 276.

204. Gretton, A., Borgwardt, K. M., Rasch, M. J., Schölkopf, B. & Smola, A. A Kernel Two-Sample Test. *Journal of Machine Learning Research* **13**, 723 (2012).

205. Fortunato, M., Blundell, C. & Vinyals, O. Bayesian recurrent neural networks. *arXiv* (2017).

206. Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A., Bridgland, A., Meyer, C., Kohl, S. A. A., Ballard, A. J., Cowie, A., Romera-Paredes, B., Nikolov, S., Jain, R., Adler, J., Back, T., Petersen, S., Reiman, D., Clancy, E., Zielinski, M., Steinegger, M., Pacholska, M., Berghammer, T., Bodenstein, S., Silver, D., Vinyals, O., Senior, A. W., Kavukcuoglu, K., Kohli, P. & Hassabis, D. Highly accurate protein structure prediction with AlphaFold. *Nature* **596** (2021).

207. Bénabou, R. & Tirole, J. Self-Confidence and Personal Motivation*. *The Quarterly Journal of Economics* **117**, 871 (2002).

208. O'reilly, J. Making predictions in a changing world—inference, uncertainty, and learning. *Frontiers in Neuroscience* **7**, 105 (2013).

209. Inglis, I. R. Review: The Central Role of Uncertainty Reduction in Determining Behaviour. *Behaviour* **137**, 1567 (2000).

210. Nalisnick, E., Matsukawa, A., Teh, Y. W. & Lakshminarayanan, B. Detecting out-of-distribution inputs to deep generative models using typicality. *arXiv* (2019).

211. Ren, J., Liu, P. J., Fertig, E., Snoek, J., Poplin, R., Depristo, M., Dillon, J. & Lakshminarayanan, B. *Likelihood Ratios for Out-of-Distribution Detection* in *Advances in Neural Information Processing Systems* (eds Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. & Garnett, R.) **32** (Curran Associates, Inc., 2019).

212. Choi, H., Jang, E. & Alemi, A. A. Waic, but why? generative ensembles for robust anomaly detection. *arXiv* (2018).

213. Kristiadi, A., Hein, M. & Hennig, P. *Being Bayesian, even just a bit, fixes overconfidence in ReLU networks* in *International Conference on Machine Learning* (2020), 5436.

214. Kristiadi, A., Hein, M. & Hennig, P. *An Infinite-Feature Extension for Bayesian ReLU Nets That Fixes Their Asymptotic Overconfidence* in *Thirty-Fifth Conference on Neural Information Processing Systems* (2021).

CURRICULUM VITAE

PERSONAL DATA

Name	Christian Henning
Date of Birth	September 05, 1991
Place of Birth	Mühlhausen, Germany
Citizen of	Germany

EDUCATION

2017 – Present	Institute of Theoretical Computer Science, ETH Zürich, Switzerland *Final degree:* PhD
2014 – 2017	Leibniz Universität Hannover Hannover, Germany *Final degree:* MSc in Computer Science
2016	Purdue University West Lafayette, USA *Semester Abroad*
2011 – 2014	Leibniz Universität Hannover Hannover, Germany *Final degree:* BSc in Computer Science
2008 – 2011	Andreas-Gordon-Schule Erfurt, Germany *Final degree:* Abitur (university entrance diploma)

PUBLICATIONS

Articles in peer-reviewed journals:

1. Henning, C. & Ewerth, R. Estimating the information gap between textual and visual representations. *International Journal of Multimedia Information Retrieval* **7**, 43 (2018).

Conference and workshop contributions:

2. Henning, C. A. & Ewerth, R. *Estimating the Information Gap between Textual and Visual Representations* in *Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval* (Association for Computing Machinery, Bucharest, Romania, 2017), 14.
3. Von Oswald, J., Henning, C., Sacramento, J. & Grewe, B. F. *Continual learning with hypernetworks* in *International Conference on Learning Representations* (2020).
4. Ehret, B., Henning, C., Cervera, M. R., Meulemans, A., von Oswald, J. & Grewe, B. F. *Continual Learning in Recurrent Neural Networks* in *International Conference on Learning Representations* (2021).
5. Oswald, J. V., Kobayashi, S., Sacramento, J., Meulemans, A., Henning, C. & Grewe, B. F. *Neural networks with late-phase weights* in *International Conference on Learning Representations* (2021).
6. Henning, C., Cervera, M. R., D'Angelo, F., von Oswald, J., Traber, R., Ehret, B., Kobayashi, S., Grewe, B. F. & Sacramento, J. *Posterior Meta-Replay for Continual Learning* in *Conference on Neural Information Processing Systems* (2021).
7. Henning, C., von Oswald, J., Sacramento, J., Surace, S. C., Pfister, J.-P. & Grewe, B. F. *Approximating the predictive distribution via adversarially-trained hypernetworks* in *NeurIPS Workshop on Bayesian Deep Learning* (2018).
8. Henning, C., D'Angelo, F. & Grewe, B. F. *Are Bayesian neural networks intrinsically good at out-of-distribution detection?* in *ICML Workshop on Uncertainty and Robustness in Deep Learning* (2021).

9. Cervera, M. R., Dätwyler, R., D'Angelo, F., Keurti, H., Grewe, B. F. & Henning, C. *Uncertainty estimation under model misspecification in neural network regression* in *NeurIPS Workshop on Robustness and misspecification in probabilistic modeling* (2021).

Preprints:

10. D'Angelo, F. & Henning, C. *Uncertainty-based out-of-distribution detection requires suitable function space priors*