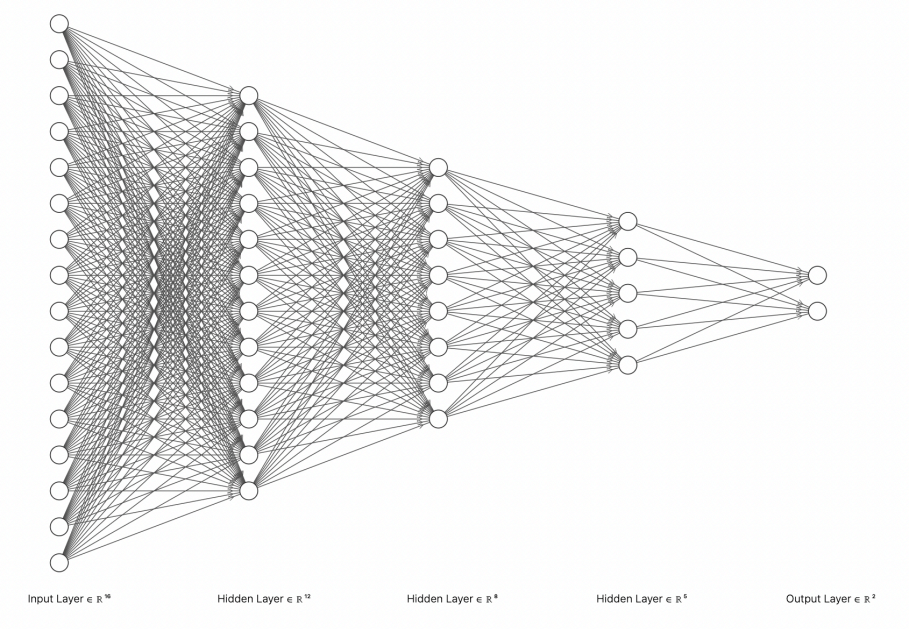
Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Engineering
Department of Mechanical and Industrial Engineering

Thomas Lunde Fosen

Study on Uncertainty Quantification in Deep Learning and its Influence on AI-Enabled Decision-Making

Master's thesis in Engineering & ICT
Supervisor: Shen Yin
Co-supervisor: Andreas Hafver
June 2023



NTNU
Norwegian University of Science and Technology

Thomas Lunde Fosen

Study on Uncertainty Quantification in Deep Learning and its Influence on AI-Enabled Decision-Making

Master's thesis in Engineering & ICT
Supervisor: Shen Yin
Co-supervisor: Andreas Hafver
June 2023

Norwegian University of Science and Technology
Faculty of Engineering
Department of Mechanical and Industrial Engineering

NTNU
Norwegian University of
Science and Technology

Uncertainty is the only certainty there is, and knowing how to live with insecurity is the only security.

John Allen Paulos

PREFACE

The following paper represents the culmination of research conducted during the spring of 2023 for the master thesis *TMM4935 - Industrial ICT*, related to the *Engineering and ICT* study program within the Department of Mechanical and Industrial Engineering at the Norwegian University of Science and Technology (NTNU). The initial project description was outlined by *DNV*, through the co-supervisor Andreas Hafver, and subsequently refined in collaboration with my supervisor, Professor Shen Yin, in order to suit my specific interests and desires for the project.

The thesis is constructed upon a theoretical foundation acquired through the course of five years of studies, and, as such, assumes a certain level of prior knowledge on the part of the reader, specifically in terms of mathematical and statistical concepts, as well as the workings of neural networks and deep learning models in general. Ergo, the intended target group can be viewed as any student or individual with some relevant scientific or academic interest and background.

Thomas Lunde Fosen Trondheim 28/05/2023

ACKNOWLEDGEMENTS

First and foremost I would like to thank my supervisor, Shen Yin, and co-supervisor, Andreas Hafver, for taking on their roles and for granting me the flexibility and autonomy to shape the thesis according to my desires, alongside providing valuable guidance. Additionally, I would like to express my gratitude to DNV for providing the initial thesis description.

ABSTRACT

Contemporary advances in machine learning and artificial intelligence have emphasized the need for reliable and efficient safety routines when implementing any such system into real-world applications, as the harm caused by erroneous predictions and inferences could have severe negative consequences. Therefore, for any autonomous system based on a deep learning framework to be viable it must incorporate some notion of confidence or uncertainty in terms of its predictions and decisions. Traditional approaches in deep learning have generally lacked this ability, which has necessitated the development of specialized methodologies and techniques explicitly designed to address and mitigate this deficiency. These concerns introduce the field of *uncertainty quantification*, which serves as the theoretical foundation of this study.

The primary objective of this thesis is to familiarize the reader with uncertainty quantification in deep learning, by presenting fundamental theoretical concepts and challenges, alongside the state-of-the-art of current methodology and techniques within the field. The thesis offers a combination of theoretical understanding and practical insights through implementation of methods, investigating how the theoretical concepts relate to real-world application. Additionally, recommendations and guidelines are presented, to offer readers general guidance. By providing a comprehensive overview of the field, the thesis aims to equip researchers, practitioners, and decision-makers with the knowledge and tools necessary to enhance the robustness, trustworthiness, and safety of machine learning systems in real-world applications.

The key findings of the study suggest that using *Deep Ensembles* with five members is a sensible starting point when quantifying uncertainty in deep learning. However, in scenarios involving transfer learning or concerns regarding the infrastructure load of hosting multiple models, an alternative approach is to employ *Monte Carlo Dropout* with a range of 30 to 100 stochastic forward passes through the network. Additionally, *Temperature Scaling* is recommended as a post-hoc technique to calibrate predictive probabilities in classification models.

SAMMENDRAG

Moderne fremskritt innen maskinlæring og kunstig intelligens har understreket behovet for pålitelige og effektive sikkerhetsrutiner knyttet til implementeringen av slike systemer i virkelighetsnære applikasjoner, da skade forårsaket av feilaktige prediksjoner og slutninger kan ha alvorlige negative konsekvenser. På bakgrunn av dette må ethvert autonomt system basert på dyp læring inkorporere en form for tillit eller usikkerhet når det gjelder prediksjoner og beslutninger for å være levedyktig. Tradisjonelle tilnærminger innen dyp læring har manglet evnen til å fange opp og representere usikkerheten knyttet til deres slutninger på en effektiv måte, noe som har nødvendiggjort utviklingen av spesialiserte metoder og teknikker som er designet for å håndtere og motvirke denne mangelen. Disse bekymringene gir opphav til feltet *usikkerhetskvantisering*, som utgjør det teoretiske grunnlaget for denne avhandlingen.

Det primære målet med oppgaven er å gjøre leseren kjent med usikkerhetskvantisering i dyp læring, ved å presentere grunnleggende teoretiske prinsipper og utfordringer, i tillegg til eksisterende metodikker og teknikker innen feltet. Oppgaven tilbyr en kombinasjon av teoretisk forståelse og praktiske innsikter gjennom implementering av metoder, for å undersøke hvordan de teoretiske konseptene relaterer seg til virkelighetsnær anvendelse. I tillegg vil anbefalinger og retningslinjer blir presentert for å gi leserne generell veiledning. Ved å gi en omfattende oversikt over feltet har oppgaven som mål å utruste forskere, praktikere og beslutningstakere med kunnskapen og verktøyene som er nødvendige for å styrke robustheten, påliteligheten og sikkerheten til maskinlæringsstystemer i praktiske applikasjoner.

De viktigste funnene i studien antyder at bruk av *Deep Ensembles* med fem medlemmer er et fornuftig utgangspunkt når man kvantifiserer usikkerhet i dyp læring. I scenarier som involverer overføringslæring eller bekymringer knyttet til infrastrukturell belastning ved hosting av flere modeller derimot, er en alternativ tilnærming å benytte seg av *Monte Carlo Dropout* med et spekter på 30 til 100 stokastiske gjennomkjøringer gjennom nettverket. I tillegg anbefales *Temperature Scaling* som en post-hoc teknikk for å kalibrere prediksjonssannsynligheter for klassifiseringsmodeller.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

AI - Artificial Intelligence

BNN - Bayesian Neural Network

DNN - Deep Neural Network

ECE - Expected Calibration Error

IID - Independent and Identically Distributed

LLM - Large Language Models

LSTM - Long-Short-Term-Memory

ML - Machine Learning

MLE - Maximum Likelihood Estimation

MSE - Mean-Squared-Error

MCD - Monte Carlo Dropout

MI - Mutual Information

NLP - Natural Language Processing

NN - Neural Network

NIG - Normal-Inverse Gamma

OOD - Out-Of-Distribution

RNN - Recurrent Neural Network

UQ - Uncertainty Quantification

CHAPTER

ONE

INTRODUCTION

1.1 Background and Motivation

Novel advancements within machine learning has led to a widespread integration of AI-systems into modern society, demonstrating remarkable performance in tasks ranging from image recognition, natural language processing and medical diagnosis. Subsequently, a demand for reliable and efficient safety-routines surrounding the use of these systems has arisen, emphasizing the importance of capturing and accurately portraying the uncertainty related to any generated decision. Therefore, an important prerequisite for any such system to be viable is an inherent awareness regarding the uncertainty of the inferences they produce, as the harm caused by potential errors could have dire consequences. Machine learning models inevitably encounter uncertainty, either stemming from the inherent uncertainty of the physical phenomena they are interacting with, or due to potentially amendable factors such as lack of data, outliers or model deficiencies. With these concerns in mind the field of *uncertainty quantification* (UQ) is introduced, as the fundamental basis of this thesis.

Formally stated, uncertainty quantification may be formalized as "*the process of quantifying uncertainties associated with model calculations of true, physical quantities of interest, with the goal of accounting for all the relevant sources of uncertainty and quantifying the contributions of specific sources to the overall uncertainty*" (Council 2012). As a consequence of the aforementioned safety concerns a growing interest in the field of UQ has been sparked, stemming from a desire to enhance the reliability of artificial models, ultimately facilitating the practical implementation of these systems in real-world applications.

The need for uncertainty-awareness in AI-enabled decision-making becomes especially evident in safety-critical applications, such as autonomous driving, medical diagnosis, finance and various socio-technical systems, where potential errors could have severe negative consequences (Varshney and Alemzadeh 2017). Accurately quantifying the associated uncertainty of an inference allows for caution to be applied, in terms of human intervention, re-evaluation, rejection, safety-thresholds, or other specific countermeasures. Essentially, the main goal is to avoid situations and scenarios in which the artificial model is confidently wrong,

meaning, the actual inference is faulty, even though the model is certain that it is correct.

Traditional deep learning classifiers have lacked the ability to accurately portray the uncertainty of their inferences, as the conventional use of the softmax function in the activation layer of neural networks make them susceptible to producing overconfident predictions and deceptive representations. The predictive probabilities obtained from these classifiers are often erroneously interpreted as model confidence (Gal 2016), which can lead to misleading judgments, especially when encountering out-of-distribution (OOD) samples. Due to inherent limitations in the mathematical foundations of the softmax activation function, these models are prone to being confidently wrong in their decisions, which emphasizes the need for alternative approaches to amend for these deficiencies.

When quantifying uncertainty in a machine learning framework a core aspect and desirable feature is the ability to distinguish and categorize underlying sources of the arising uncertainty, mainly in terms of so-called *epistemic* (model) and *aleatoric* (data) uncertainty. The nature of the arising uncertainty has distinct implications in terms of counteracting and engineering an accurate response, which makes the ability to accurately identify the source valuable. This relates directly to another key feature, namely detecting whether a data sample does not belong to the previously observed data distribution, otherwise known as out-of-distribution samples, as systems operating in real-world applications are prone to being exposed to shifts in the data distribution.

Traditional approaches for quantifying uncertainty often rely on statistical methods, specifically Bayesian models that involve learning a statistical distribution over model weights. The computational complexity of these approaches can be prohibiting, however, especially when dealing with complex models and data. To address these challenges and explore alternative means of modeling uncertainty, researchers have proposed various alternative avenues of research. These novel approaches aim to broaden the range of techniques available for uncertainty quantification and overcome the computational limitations associated with traditional methods. By investigating alternative modeling strategies, the goal is to enhance the accuracy, efficiency, and scalability of uncertainty quantification, in diverse applications within the field of deep learning.

1.2 Project Description

The main purpose of the thesis is to serve as a valuable resource for both academic and industrial pursuits in the field of uncertainty quantification in deep learning. The primary objective is to provide a comprehensive overview of the necessary theoretical foundations and existing methods, presenting the state-of-the-art as described in contemporary literature. Furthermore, the thesis includes a series of experiments conducted on benchmark datasets to investigate and explore the practical implementation of these methods. By offering this combination of

theoretical understanding and practical insights, the thesis aims to introduce the reader to available methodologies and frameworks while providing guidance for their application. Additionally, the thesis aims to identify and highlight potential limitations and future research directions within the field, equipping the reader with the knowledge needed to address these challenges.

The thesis is primarily focused on the engineering and technical aspects related to the field, as opposed to the ethical, social and safety related concerns surrounding the use of AI-systems. Instead, it centers on higher-order theoretical concepts, including some that may be considered esoteric. Therefore, a certain level of prior knowledge is assumed on the part of the reader, particularly in mathematical and statistical foundations, as well as a general understanding of neural networks. The thesis aims to provide a deep exploration of these concepts, catering to readers who possess the necessary background to engage with the intricate technical aspects of uncertainty quantification in deep learning.

1.2.1 Objectives

The thesis can be divided into three distinct sub-objectives, each contributing to its overall objective. These sub-objectives can be explicitly stated as follows:

1. The first sub-objective is to provide a comprehensive introduction to the field of uncertainty quantification. This involves exploring and presenting the necessary theoretical foundations, ensuring the reader develops a strong understanding of the fundamental concepts.

2. The second sub-objective is to present specific methods from the existing literature, aiming to provide the reader with an overview of available methodologies, enabling them to navigate the field effectively. Additionally, this exploration will help identify potential gaps and challenges that can guide future research endeavors.

3. The third sub-objective involves conducting experiments on benchmark datasets, aiming to bridge the gap between theoretical foundations and practical implementation. By investigating the practical implications of the theoretical concepts, the thesis aims to enhance the reader's understanding of how uncertainty quantification methods operate in practical settings. The specifics of the experiments are introduced in Section 4

By achieving these three sub-objectives, the thesis aims to fulfill its broader objective of providing a thorough understanding of uncertainty quantification in deep learning, equipping the reader with theoretical knowledge, practical guidelines, and the ability to partake and contribute to this evolving field. Two things to note is that throughout the course of the thesis, the terms *artificial intelligence*, *machine learning* and *deep learning* will be used somewhat interchangeably depending on the context, essentially referring to the same concept, with a varying degree of generality. Also, in terms of the scope, the thesis is mainly concerned with classifiers, where as regression models will be included whenever suitable and relevant.

1.3 Related Work

A continually expanding body of literature exists following the recent surge in development and relevancy of AI-systems. Recognizing and acknowledging the pioneering authors is essential, however, as their breakthroughs have laid the groundwork and paved the way for subsequent and future innovations. The following section aims to provide a concise overview of the primary authors and their impact and contributions to the broader field, while a comprehensive review and discussion surrounding their work will be presented in subsequent sections. Otherwise, supplementary literature and sources will be explicitly referenced when deemed fit, to maintain the integrity of the presented material. Furthermore, parts of the thesis extend upon the corresponding specialization project conducted during the autumn of 2022 for the course *TPK4550 - Reliability, Availability, Maintainability and Safety* at the RAMS department of NTNU. Consequently, although all material has been revised, certain sections might appear overlapping or inspired by the specialization report. In these parts, the initial authors are fully acknowledged and credited for their contributions.

Gal and Ghahramani (2016) first introduced *Monte Carlo Dropout* as a means of quantifying uncertainty in deep learning models, which has since gained significant recognition in the field. Building upon this foundation, Lakshminarayanan, Pritzel and Blundell (2016) extended the same theoretical concept to ensemble methods, resulting in the development of *Deep Ensembles*. These two methodologies have established themselves as reliable and efficient approaches, forming the basis of the practical work undertaken in this thesis. In a somewhat distinct direction from the underlying principles of the previously mentioned methods, two novel approaches were introduced for approximating uncertainty in neural networks. The first framework, known as *Prior Networks*, was proposed by Malinin and Gales (2018), while another framework built on similar foundations, called *Evidential Deep Learning*, was proposed by Sensoy, Kaplan and Kandemir (2018) and Amini et al. (2019). Numerous avenues and fields of research explore new methods for uncertainty quantification, however, given the scope of the thesis, these two approaches were selected as alternative approaches to the previously recognized methods.

In addition to the aforementioned methods that explicitly quantify uncertainty, another approach known as *Temperature Scaling* is included. The method is proposed by Guo et al. (2017), and focuses on calibrating the generated predictions of a classification model. While its intended functionality differs from the previously discussed methods, it is considered a valuable supplement for any classification model involved in prediction generation. Therefore, the method is included as a useful addition to the repertoire of uncertainty quantification techniques.

1.4 Thesis Structure

The thesis follows a structured framework outlined as follows:

Section 1 introduces the background and motivation for the thesis, and specifies the project description.

Section 2 provides a thorough presentation of the necessary theoretical prerequisites and foundations of uncertainty quantification within deep learning, and aims to establish the required background knowledge.

Section 3 explores existing methods from the literature, thus providing an overview of various methodologies employed in uncertainty quantification.

Section 4 focuses on practical experiments conducted to investigate and validate some of the presented methods. The results obtained from these experiments are analyzed and discussed.

Section 5 provides a comprehensive discussion of the findings, highlighting the implications, limitations, and potential future directions in uncertainty quantification research.

Section 6 concludes the thesis by summarizing the key findings and contributions.

CHAPTER

TWO

THEORY

The upcoming section aims to introduce and review the underlying theoretical principles of uncertainty quantification in deep learning, serving as a conceptual foundation for the corresponding methodology. As mentioned, although an introduction to certain basic concepts will be provided, several of the concepts are potentially esoteric, and, as such, some prior knowledge is assumed on the part of the reader, specifically in terms of mathematical and statistical concepts, as well as the basics of deep learning and neural networks.

2.1 Uncertainty Quantification

An important prerequisite for any viable machine learning system intended to make decisions based on real-world phenomena is a trustworthy representation of the uncertainty and confidence of a generated inference. As such, reliable methods and systems for accurately capturing and portraying these uncertainties are necessary, based on the underlying theoretical field of uncertainty quantification, which will be introduced in this section.

In addition to safety-critical concerns, it is worth mentioning that quantification of uncertainty also has other fields of application within deep learning, related to optimization in active learning (Aggarwal et al. 2014; Nguyen, Destercke and Hüllermeier 2019), and within certain specific learning algorithms, such as decision trees (Tom Michael Mitchell 2002).

Generally, the main sources of uncertainty in machine learning modeling include, but are not limited to (Gal 2016);

i) *Out-of-distribution samples* - OOD samples refer to data points that do not belong to the initially observed and modeled data distribution. This generally renders the model unable to accurately classify and recognize the sample, unless the model has been explicitly designed to include features that can capture these shifts in the data distribution.

ii) *Data noise* - The observable data might contain inherent noise or inaccuracies, which is traditionally referred to as *aleatoric uncertainty.*

iii) *Uncertainty related to model parameters and structure* - Uncertainty related to the choice of model structure and corresponding parameters that most accurately portrays the data can arise, as oftentimes there exists a wide range of potential models that fit the data. Also, scarcity of available data points contribute to this uncertainty. This makes up what is referred to as *epistemic uncertainty*.

Traditionally, methods for capturing uncertainty in machine learning models have yet to be concerned with distinguishing between the underlying sources of the arising uncertainty. In recent years, however, efforts have been made in order to explore the utility and viability of developing means of distinguishing the sources, especially in terms of aleatoric and epistemic uncertainty, as this would enable developers to more accurately produce adequate countermeasures for handling the uncertainty (Hüllermeier and Waegeman 2021).

On the basis of these descriptions, an introduction to aleatoric and epistemic uncertainty is necessary, as these sources constitute the majority of the uncertainty related to machine learning models and play a large role in many of the contemporary advances within the field.

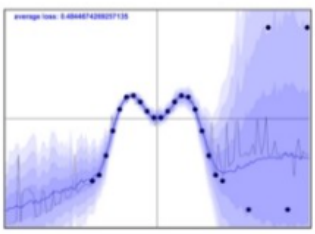
2.1.1 Aleatoric versus Epistemic Uncertainty

A crucial distinction to make when evaluating uncertainty is whether the underlying source is of *aleatoric* or *epistemic* nature, as this allows for accurately implementing specific countermeasures (Hüllermeier and Waegeman 2021). Aleatoric uncertainty, often called *data uncertainty*, generally stems from the inherent uncertainty of studied phenomena, characterized by being *irreducible*. The uncertainty being irreducible means that no additional information exists, and no possible improvements to the developed model can be made, to account for and reduce the uncertainty. A textbook example of aleatoric uncertainty is that of a coin-toss. The non-deterministic nature of this phenomenon excludes the possibility of producing and gaining further insight into the event's outcome, thus deeming the uncertainty *irreducible*. Generally, this underlying source of uncertainty is encountered in terms of noisy, imprecise or low-resolution data.

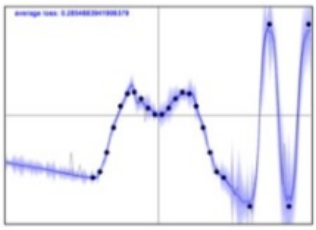
In addition to this categorization a further distinction can be made between *homoscedastic* and *heteroscedastic* aleatoric uncertainty, usually described as *constant* and *variable* data uncertainty, respectively. The key distinction between the two is that homoscedastic uncertainty is invariant to input samples, meaning the uncertainty is constant across the entire observation space and noise does not vary across inputs. This representation is rarely accurate when operating with real-world phenomena and related data spaces, however, making it less useful for a viable implementation.

Heteroscedastic uncertainty, on the other hand, refers to scenarios in which the inherent randomness of data is a function of the data itself, meaning the level of observational noise varies across the data distribution (Davis, Zhu and Oldfather 2020). The observed noise across a data space relating to real-world phenomena

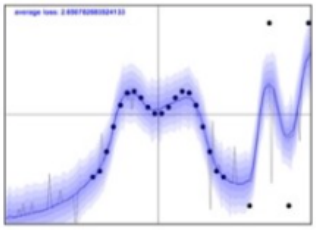
tends to vary, which makes the heteroscedastic representation of uncertainty better suited for describing and handling these processes. Illustrations of the aforementioned sources of uncertainty are presented by Gal (2015), in Figure 2.1.1.



(a) Heteroscedastic Uncertainty



(b) Low Homoscedastic Uncertainty



(c) High Homoscedastic Uncertainty

Figure 2.1.1: Observational noise visualized through three different uncertainty representations, presented by Gal (2015). In (a) the data uncertainty, represented by the blue shadings, varies across the input space. Specifically, the uncertainty is high in areas containing few data samples, whereas in areas containing many data samples, the uncertainty is low. This is generally the most accurate and desired portrayal of a system modeling real-world phenomena, referred to as *heteroscedastic uncertainty*. The uncertainty in (b) and (c), on the other hand, does not vary but is constant across the input space and has a constant, finite, variance. It is evident from these models that they do not accurately reflect data noise, outliers, and lack of data, making this representation less viable as a method for describing physical phenomena. This is referred to as *homoscedastic uncertainty*.

In contrast to aleatoric uncertainty, epistemic, or *knowledge uncertainty*, refers to uncertainty that stems from a lack of information and insufficient knowledge. Due to this fact, this uncertainty is *reducible*, as it can in principle be reduced by additional knowledge, improving the model's ignorance and inaccuracy. Generally, lack of training data is a major source of epistemic uncertainty, one example being the under-representation of racial minorities in a facial recognition dataset, or the absence of rare words in a language representation model. In this scenario, countermeasures to improve and reduce the arising uncertainty can be made, which makes the distinction and classification of the uncertainty useful. A visualization of aleatoric and epistemic uncertainty in a linear process is illustrated in Figure 2.1.2.

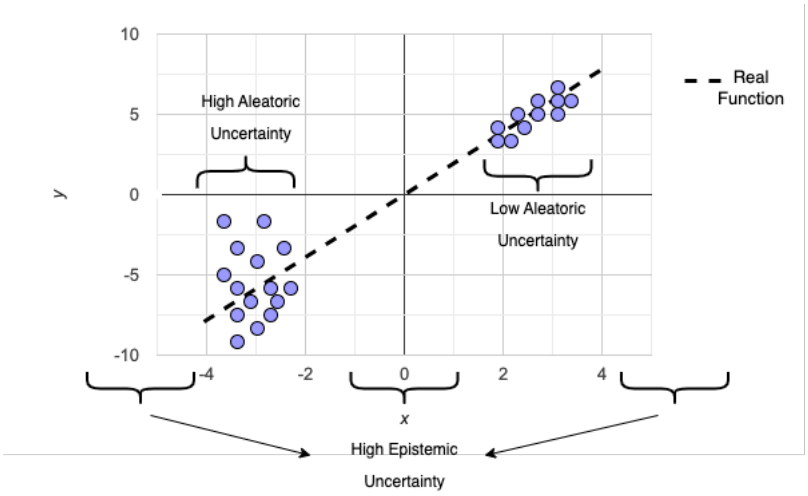


Figure 2.1.2: Visualization of *aleatoric* and *epistemic* uncertainty in a linear process. In the areas where the model lacks data points, epistemic uncertainty arises, as the model does not contain sufficient information to produce reliable and confident inferences. In the areas containing enough data, however, the uncertainty arises from the inherent variance of the data itself, as distinguished in the left and right cluster of data points in the linear process.

Identifying the underlying sources of uncertainty as either aleatoric or epistemic entails identifying whether the uncertainty is reducible or not, allowing for potential specific countermeasures to be implemented. As of yet, traditional machine learning methods have not been equipped to explicitly capture this distinction, or any measure of uncertainty to a great extent, as they are mostly based on purely statistical approaches. However, as machine learning and artificial models are becoming more and more extensive and prevalent in modern technology, the need for novel approaches for capturing uncertainty and accurately identifying the underlying sources has arisen. Kendall and Gal (2017) show that capturing and identifying epistemic and heteroscedastic aleatoric uncertainties independently has a positive impact on the predictive performance of models, and that combining the two improves the performance even further.

The team presented further insight, namely that modeling aleatoric uncertainty is highly advantageous for real-time applications and when large amounts of data are available. For safety-critical applications and when there is a general lack of data, they concluded that modeling epistemic uncertainty is vital, especially if the model needs to be aware of potential out-of-distribution samples. The team produced state-of-the-art performance on benchmark tasks within semantic segmentation and depth regression, by realizing that the two sources of uncertainty are not mutually exclusive, but can be combined, thus proving the viability and utility of capturing the aforementioned sources of uncertainty.

2.1.2 Mathematical Representations

To accurately model uncertainty in a desired deep learning framework, a mathematical representation of the relevant concepts is useful, as a guideline, in addition to providing a theoretical foundation. Such a representation is presented by Hüllermeier and Waegeman (2021) in combination with Abdar et al. (2021), whose formulations will be introduced in this section.

As a starting point, the total *predictive uncertainty* is presented as a baseline for further derivations, as the sum of aleatoric and epistemic uncertainty, yielding

$$\underbrace{PU}_{Predictive} = \underbrace{AU}_{Aleatoric} + \underbrace{EU}_{Epistemic} \tag{2.1}$$

Considering a set of training data $\mathcal{D} := \{(x_1, y_1), \ldots, (x_N, y_N)\} \subset X \times Y$ for an instance space X and a set Y comprised of potential outcomes associated to an instance, the non-deterministic dependency between the two produces the *conditional probability distribution* on Y, as stated in Equation (2.2). The dependency is described through new instances, x_i, in the instance space.

$$p(y|x_i) = \frac{p(x_i|y)}{p(x_i)} \tag{2.2}$$

The conditional probability distribution on Y, referred to as the *predictive posterior distribution* within Bayesian inference, holds the property that in terms of aleatoric uncertainty, even given complete information regarding the density p, uncertainty related to the outcome of y remains, as it does not uniquely identify a single outcome of y.

Generally, assuming a *hypothesis space* H, a learner is concerned with generating a hypothesis $h^* \in \mathcal{H}$ that minimizes the expected loss, given the loss function $\ell : Y \times Y \to \mathbb{R}$, where the generated hypotheses are mappings between the instances and outcomes, namely $h : X \to Y$. To minimize the expected loss, the *point-wise Bayes predictor*, as given in Equation (2.3), provides the most accurate predictions.

$$f^*(x) := \underset{\hat{y} \in Y}{argmin} \int_Y \ell(y, \hat{y})\, dP(y|x) \tag{2.3}$$

The deviation between the point-wise Bayes predictor and the *true risk minimizer*, as stated in Equation (2.4), is directly correlated to the epistemic uncertainty due to the fact that it represents the correctness of the model and choice of hypothesis space H.

$$h* := \underset{h \in H}{argmin}\, R(h) \tag{2.4}$$

Then, given a training dataset comprised of C classes, as well as a function $y = f^w(x)$ parameterized by w, the goal is to optimize said parameters, aiming to fit the function to the desired outputs, where the epistemic uncertainty is then represented as the probability distribution over these parameters. Usually, within

Bayesian statistics, this is done by using the softmax likelihood for classification tasks, and the Gaussian likelihood for regression tasks (Abdar et al. 2021). These are formulated in Equations (2.5) and (2.6), respectively.

$$p(y|X,w) = \frac{e^{f_y^w(X)}}{\Sigma_{i=1}^{C} e^{f_{y_i}^w(X)}} \tag{2.5}$$

The softmax likelihood for training data comprised of C classes, model parameters w and inputs X is given in Equation (2.5), whereas the Gaussian likelihood for the same scenario is given in Equation (2.6), where τ refers to the model precision, and I is the identity matrix.

$$p(y|X,w) = \mathcal{N}(y; f^w(x), \tau^{-1}\, I) \tag{2.6}$$

The aleatoric uncertainty from the model can then be captured by formulating the model precision as a function of the data. Several approaches exist for capturing the corresponding epistemic uncertainty, one of which being based on combining two functions, namely the predictive mean, $f^\theta(x)$, and the model precision, $g^\theta(x)$. By doing so, the likelihood function can be defined as $y_i = \mathcal{N}(f^\theta(x), g^\theta(x)^{-1})$, which can then be combined with the Euclidean distance loss function, formulated in Equation (2.7), as a means of representing the predictive variance, as derived in Equation (2.8)

$$E^{W_1,W_2,b}(X,Y) = \frac{1}{2N}\Sigma_{i=1}^{N} \, ||y_i - \hat{y}||^2 \tag{2.7}$$

Equation (2.7) states the Euclidean distance loss function, whereas Equation (2.8) defines the predictive variance, representing epistemic uncertainty.

$$\hat{Var}[X*] := \frac{1}{T}\Sigma_{t=1}^{T} g^{\tilde{w}_t}(X) I + f^{\tilde{w}_t}(X^*)^T f^{\tilde{w}_t}(X^*) - \tilde{E}[y^*]^T \tilde{E}[y^*] \underset{T\to\infty}{\to} Var_{q_\theta^*(y^*|x^*)}[y^*] \tag{2.8}$$

Several approaches and methods for representing and capturing the aforementioned sources of uncertainty exist, with the presented framework being intended to serve as a basis for gaining insight into the necessary theoretical foundations of any such approach.

2.1.3 Decomposing Uncertainty in a ML Framework

Based on the presented mathematical foundations, a notational framework for representing the decomposition of total uncertainty within a machine learning framework can be introduced. Specifically, a formalization of the terms representing uncertainty within a Bayesian framework will be presented, as statistical approaches constitute the foundation of many relevant methodologies within the field. This formalization is provided by Davis, Zhu and Oldfather (2020).

Generally, Bayesian neural networks (BNN) are concerned with generating a set of predictions, from which a predictive distribution can be used in order to estimate the variance of said distribution over predictions. By use of the total law of

variance, also known as *Eve's law*, the variance of this distribution, which can be viewed as a measure of the total uncertainty of the model, can be decomposed as stated in Equation (2.9) (Kendall and Gal 2017). Considering a BNN with random model parameters θ, an input sample x with corresponding target variable y, and the expected value $E()$, the following describes the relationship between the total uncertainty, the expected value of a target variable, and its corresponding variance.

$$V(y|x) = V(\underbrace{E[y|x,\theta]}_{f(x,\theta)}) + E[\underbrace{V(y|x,\theta)}_{\sigma^2(x,\theta)}] \tag{2.9}$$

Equation (2.9) describes the relationship between the expected value of a target variable y and its corresponding variance. The expected value of the target variable is represented as $E[y|x,\theta])$, which stems from the definition as defined in Equation (2.10). The corresponding variance, $V(y|x,\theta)$, is stated as $\sigma^2(x,\theta)$, and defined in Equation (2.11).

$$f(x,\theta) = E[y|x,\theta] \tag{2.10}$$

The expected value of a target variable can be estimated by a single forward pass through the neural network, from which the corresponding variance can also be obtained.

$$\sigma^2(x,\theta) = V(y|x,\theta) \tag{2.11}$$

Following these properties, the variance related to the BNN's predicted mean, $V(f(x,\theta))$, and the average of the BNN's predicted variance, $E[s^2(x,\theta)]$, constitute the total uncertainty, as

$$\textit{Total Uncertainty} = \underbrace{V(f(x,\theta))}_{\textit{Epistemic Uncertainty}} + \underbrace{E[\sigma^2(x,\theta)]}_{\textit{Aleatoric Uncertainty}} \tag{2.12}$$

Based on the aforementioned properties, Kendall and Gal (2017) present a set through which the total uncertainty of a model can be obtained, stated in Equation (2.13).

$$\{f(x,\theta_t), \sigma^2(x,\theta_t)\}_{t=1}^{T} \sim BNN_{\theta_t}(x) \text{ where } \theta_t \sim p(\theta|\mathcal{D}) \tag{2.13}$$

This equation formalizes a set comprised of the mean and variance obtained from a single forward propagation through the network for a fixed input sample x, repeated T distinct times to generate a set. θ_t refers to the random model parameters for each distinct forward pass, where $\sim$ refers to a single simulation

of the BNN. Then, the epistemic and aleatoric uncertainties can be obtained from samples of the set, either separately or conjoined, following the property stated in Equation (2.9).

Though less prevalent in existing methods, these formulations can also be altered to distinctly capture heteroscedastic aleatoric uncertainty, as formulated in Equation (2.14), by averaging a sample set of the predictive variance.

$$\sigma^2(x) = E_{\theta \sim q(\theta,\mathcal{D})}[\sigma^2(x,\theta)] = \frac{1}{T}\sigma^2(x,\theta_t) \tag{2.14}$$

Once again, the BNN is used to estimate $\sigma^2(x,\theta)$. Ideally, if this term tends toward zero, no inherent uncertainty exists. As homoscedastic aleatoric uncertainty generally fails to accurately portray any real-world phenomena, formulations regarding this source is deemed unnecessary to introduce.

2.2 Theoretical Prerequisites

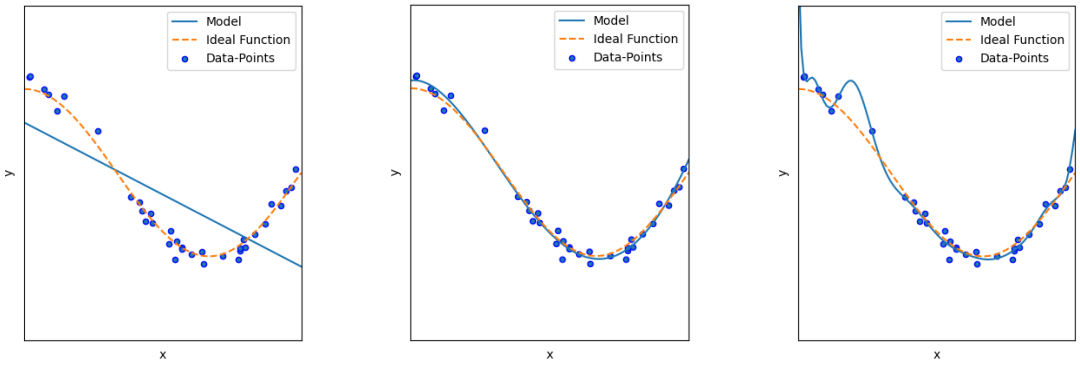
In addition to the presented theory regarding uncertainty quantification, some theoretical prerequisites are necessary to fully comprehend the methodologies and discussions to be further presented. The following section is intended to provide the reader with an introduction and overview of these concepts.

2.2.1 Dropout

As dropout constitutes the theoretical foundation of one of the major contemporary methods within the field, namely *Monte Carlo Dropout*, a brief introduction to the theoretical underpinnings of this feature should be presented in order to provide a thorough understanding of the mechanism of the method.

Initially, dropout was introduced by Srivastava et al. (2014) as a means of preventing *overfitting* in complex deep neural networks, by randomly dropping certain neurons and corresponding connections from the network during training, essentially preventing neurons from co-adapting too much, ensuring that the model does not become overly dependent on the training data. Overfitting refers to scenarios where a machine learning model erroneously learns the noise and specificities of a dataset, instead of generalizing to the underlying patterns and relationships describing said data. In layman's terms, this means that the model learns to merely recognize the specific data samples contained in the dataset, as opposed to learning the features that describe said samples, in order to be able to recognize future, unseen samples of a similar class. As opposed to this, *underfitting* refers to scenarios in which the model is not able to learn the descriptive features of the given data.

An illustration of overfitting and underfitting is presented in Figure 2.2.1. By looking at a collection of data samples, represented in the graphs by blue dots, the estimated regression function of the machine learning model, presented as the light blue line, either becomes too heavily dependent on the specific samples, as is evident in 2.2.1c, or fails to learn any features from the data, as shown in 2.2.1a. Overfitting could occur due to a number of reasons, with the most common being an insufficient amount of training data, or excessive training of the model.



(a) Underfitting **(b)** Well calibrated fit **(c)** Overfitting

Figure 2.2.1: Graphs visualizing *overfitting* and *underfitting*. The graphs show the estimated regression function of a machine leaning model, in light blue, and the ideal function describing the data samples, as the dotted line. (a) presents a scenario where the model is unable to learn any features from the data samples, and merely describes the data by means of a simplistic linear model, which is referred to as *underfitting* the model. (c), on the other hand, visualizes a scenario in which the model has become too heavily dependent on the specific data points, and thus learns the specific samples, making it less useful when introducing new samples. This is what is referred to as *overfitting*. (b) visualizes a well calibrated fit, where the model accurately captures the trends of the observed data, instead of the specific data samples.

Srivastava et al. (2014) provide an illustration of the general structure of dropout, in Figure 2.2.2. The illustration shows a neural network before and after dropout is applied, where certain neurons and corresponding connections are dropped depending on the specified *dropout rate*, which, essentially, is a measure of the probability of a single neuron being dropped.

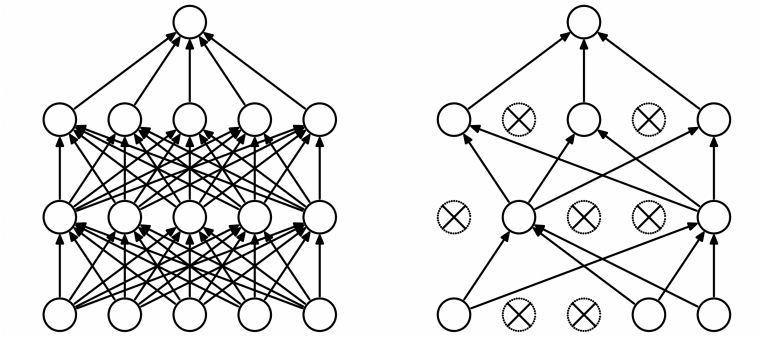


Figure 2.2.2: Model architecture of a neural network before (left) and after (right) dropout is employed, presented by Srivastava et al. (2014). As shown in the architecture in the right model, certain neurons are switched off in each layer of the network, dependent on the *dropout rate*. By doing so, the neural networks avoids becoming overly dependent on the observed data samples, as the model is forced to generalize through removing certain connections in the network.

How this regularization technique can be utilized as a means to capture model uncertainty is presented and discussed in Section 3.1.

2.2.2 Deficiencies of the Softmax Probability

A crucial feature for any machine learning model concerned with producing inferences in real-world applications is the ability to recognize and distinguish input samples that do not belong to the previously observed and modeled data, formally known as *out-of-distribution* (OOD) samples. Traditionally, the use of the softmax activation function has been prevalent within multi-class classification (Sensoy, Kaplan and Kandemir 2018), which has entailed certain limitations in terms of interpreting the outputs of the models when encountering OOD samples, some of which will be discussed in the following section.

The central issue that arises in terms of uncertainty quantification when the softmax activation function is employed in the output layer of a neural network is that the resulting prediction probabilities of an inference are limited to the already modeled classes of data, meaning, the output is purely a measure of the probability of the input sample belonging to one of the modeled classes, as compared to the other modeled classes. In other words, the softmax probabilities inherently lack any means of providing insight into whether an input sample actually belongs to the observed data distribution, and therefore, due to the function's mathematical properties and limitations, the generated probabilities are merely applicable for conclusions regarding their comparative value against the specified classes, and do not represent general model confidence.

As such, the resulting prediction probabilities are often erroneously interpreted as model confidence (Gal and Ghahramani 2016), even though a high softmax probability does not necessarily correlate to high prediction certainty, as the output from the softmax layer can merely be interpreted as a probability vector over the modeled classes. On the basis of the aforementioned limitation, an introduction to the functionality and underlying theory of the softmax activation function should be presented, in order to provide the reader with an understanding of the mechanisms that need to be accounted for in order to counteract this deficiency.

Formally, the softmax activation function converts a vector of N real numbers to a probability distribution over N possible outcomes, essentially classes, as defined in Equation (2.15).

$$\sigma(U)_i = \frac{e^{u_i}}{\Sigma_{j=1}^{N} e^{u_j}} \tag{2.15}$$

Mathematically, the softmax function takes in an input vector U with elements u_i for $i = \{1, \ldots, N\}$ relating to N classes, and produces a corresponding output vector. The normalization term $\Sigma_{j=1}^{N} e^{u_j}$ ensures that the output range is fixed from 0 to 1, and that all the output elements sum to 1, essentially producing a probability distribution over the output classes. This output vector, as mentioned, is easily misconstrued as a probability vector representing model confidence, due to its probabilistic traits. The vector, however, merely normalizes the input vector, as seen in the example calculation in Equation (2.16), by use of the normalization term. Evaluating the formula, as provided in Equation (2.15), provides insight into the mechanism of the function, and how it can be viewed as a normalization

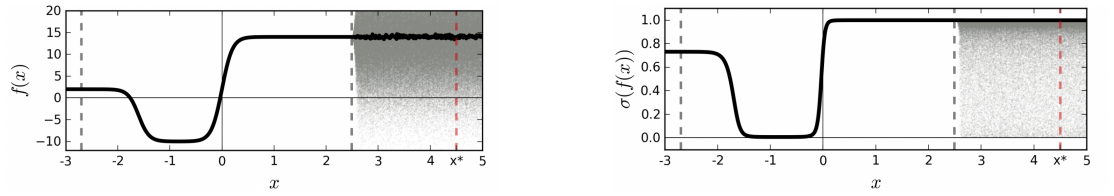
term.

The values in the output vector are essentially point estimates of each class probability for a given input sample, which, as mentioned, do not provide any conclusion regarding the associated uncertainty of the generated prediction. Equation (2.16) shows an example calculation of the softmax function, taking in one vector and outputting a normalized vector.

$$\sigma\left(\begin{bmatrix}3.2\\1.3\\0.2\\0.8\end{bmatrix}\right) \rightarrow \frac{e^{u_i}}{\Sigma_{j=1}^{J} e^{u_j}} \rightarrow \begin{bmatrix}0.775\\0.116\\0.039\\0.070\end{bmatrix} \tag{2.16}$$

An example of the softmax activation function producing an unjustifiably confident prediction is presented by Gal and Ghahramani (2016), by means of a idealized binary classification problem, illustrated in Figure 2.2.3.

Essentially, what the illustration depicts, is a scenario in which the softmax output produces an erroneously confident prediction, as the softmax function does not adjust the predictive probabilities accurately in the extrapolations of the point estimates away from the observed data. The softmax output mistakenly assigns a predictive probability of 1 to the sample x^*, represented by the dotted red line, even though the point lies far outside the observed data, which lies between the two dotted gray lines. This emphasizes how, even though the softmax output is high, the model is actually uncertain in its prediction. This can be somewhat combated, however, by instead passing the distribution as opposed to the function though the softmax activation function, as indicated by the gray shadings, where the darker shades represent higher uncertainty estimates.



(a) Softmax Input **(b)** Corresponding Softmax Output

Figure 2.2.3: Visualization of an arbitrary function $f(x)$ as softmax input in (a), with corresponding softmax output $\sigma(f(x))$ in (b), both represented as functions of the data x. The training-data exists between the two dotted gray lines, while the solid line represents the function point estimate. The shaded gray area is used to illustrate the function uncertainty, while the point x^* represents a sample that occurs far outside the observed training data. The figures are developed and presented by Gal and Ghahramani (2016).

Sensoy, Kaplan and Kandemir (2018) provide concrete formulations of the deficiencies of modeling class probabilities with the softmax activation function. Assuming a K- class classification task they define the likelihood function as Equation (2.17), for an observed tuple (x, y) comprised of a sample and its corresponding label. $\sigma()$ refers to the the softmax activation function, as presented in Equation

(2.15), and $f_j(x, \theta)$ refers to the $j-th$ output channel of the neural network $f(\cdot)$, with model parameters θ. $Mult()$ refers to a multinomial mass function that describes the probability distribution of a multinomial random variable.

$$Pr(y|x, \theta) = Mult(y|\sigma(f_1(x, \theta)), \cdots, \sigma(f_K(x, \theta))), \tag{2.17}$$

By maximizing the multinomial likelihood in terms of the model parameters, the softmax function can reduce the resulting class probabilities into a simplex. Due to computational concerns, however, the negative log-likelihood, formally known as the *cross entropy loss*, is generally minimized instead. This loss is formulated as

$$-log\, p(y|x, \theta) = -log\, \sigma(f_y(x, \theta)) \tag{2.18}$$

The negative log-likelihood, as stated in Equation (2.18), can be viewed as maximum likelihood estimation when interpreted probabilistically, which inherently is not capable of inferring predictive distribution variance, as it is a frequentist technique. Another issue is that the predictions generated based on the softmax activation function are prone to being exaggerated by outliers, due to the function's exponent on the neural network's outputs. Therefore, the predictions obtained from the softmax activation layer are not useful for drawing conclusions regarding the overall confidence of an inference, other than it's comparative value against the other classes. On account of this, no information regarding the data distribution is available. An experiment performed to demonstrate this scenario will be presented in Section 4.3.

The inadequacies of the softmax output layer have produced certain challenges in terms of quantifying model uncertainty, as presented in this section. Therefore, specific features and methods are needed in order to amend for these deficiencies, some of which will be presented during the course of this thesis.

2.2.3 Bayesian Inference

A vast amount of the available methodologies within the field of uncertainty quantification are to some degree related to and/or dependent on an underlying Bayesian framework, meaning, an introduction to *Bayesian inference* is necessary in order to fully comprehend the theoretical foundations constituting these methods. Gelman et al. (2014) provide a thorough review and introduction to Bayesian inference, whose formulations will be presented in this section.

Essentially, Bayesian statistical frameworks are concerned with updating and refining beliefs about an unknown variable or parameter based on the introduction of novel information. Two core aspects of this framework that are especially important as they appear in most, if not all, statistical uncertainty quantification methods, are *prior* and *posterior distributions.*

Simply put, a prior distribution is a probability distribution used to describe the parameters of a model prior to any data or information being observed. An example of this could be to assume the prior distribution of the regression coefficients in a linear regression model to be Gaussian. In relation to this, the posterior distribution refers to the probability distribution that describes the updated prior distribution, after observing new data or information. Generally, this update from the prior to posterior distribution is calculated by multiplying a likelihood function, which essentially measures the probability of observing the data given specific parameters, with the prior distribution, by means of Bayes' theorem. One could say that the posterior distribution updates the beliefs regarding model parameters modeled in the prior distribution, based on new information.

Mathematical formulations regarding this framework can be made, following the notation presented by Gelman et al. (2014). As mentioned, generally speaking, the underlying idea behind Bayesian statistics is based on updating beliefs based on the introduction of new information. Specifically, this can be formulated in terms of *probability statements* of a parameter θ, or unobserved data $\tilde{y}$, which are conditional on the observed value of a sample y, stated as $p(\theta|y)$ or $p(\tilde{y}|y)$. The notation $p(\cdot|\cdot)$ will be used frequently, denoting the aforementioned probability statements, namely the *conditional probability density*, where the arguments are context dependent. In addition to this, the terms *distribution* and *density* will be used interchangeably, referring to the same concept.

A prerequisite for making probability statements regarding a parameter θ provided data y is a so-called *joint probability distribution* for these θ and y, which is denoted in Equation (2.19). Here, the term $p(\theta)$ refers to the prior distribution, as introduced, and the term $p(y|\theta)$ refers to the *data distribution*.

$$\underbrace{p(\theta, y)}_{\textit{Joint Probability Distribution}} = \underbrace{p(\theta)}_{\textit{Prior Distribution}} \times \underbrace{p(y|\theta)}_{\textit{Data Distribution}} \tag{2.19}$$

Then, by *conditioning*, which will be introduced in detail in Section 2.2.4, on the observed value of y by use of Bayes' rule, the prior distribution can be described as in Equation (2.20).

$$p(\theta|y) = \frac{p(\theta, y)}{p(y)} = \frac{p(\theta)p(y|\theta)}{p(y)} \tag{2.20}$$

In this context, the term $p(y)$ refers to the sum $\Sigma_\theta \, p(\theta) \, p(y|\theta)$ over all values of θ. Making inferences past observed data, often called *predictive inference*, is a core aspect of Bayesian statistics, which, following the presented notations, can be formulated as

$$p(y) = \int p(y, \theta) d\theta = \int p(\theta)p(y|\theta) \, d\theta \tag{2.21}$$

This statement can be viewed as the *prior predictive distribution* of y, as it is independent of any previous observations. Then, after observing new information, in terms of data y, an unknown observable $\tilde{y}$ can be predicted from the same process, which yields the distribution of said $\tilde{y}$ as stated in Equation (2.22). This

distribution is referred to as the *posterior predictive distribution*, as it is conditioned on observations of data.

$$\begin{aligned} p(\tilde{y}|y) &= \int p(\tilde{y}, \theta|y)\, d\theta \\ &= \int p(\tilde{y}|\theta, y)p(\theta|y)\, d\theta \\ &= \int p(\tilde{y}|\theta)p(\theta|y)\, d\theta \end{aligned} \tag{2.22}$$

The posterior predictive distribution, as described in the above equation, can be presented in terms of an average of conditional predictions over the parameter's posterior distribution, which is the case in the second and third line of the statement in Equation (2.22).

As mentioned, the posterior distribution can be attained through the likelihood function being multiplied with the prior distribution, as is the basis of the Bayes' theorem. As such, the ratio of the posterior density, at points θ_1 and θ_2, produces what is referred to as the *posterior odds* for θ_1 as compared to θ_2, formally stated in Equation (2.23).

$$\frac{p(\theta_1|y)}{p(\theta_2|y)} = \frac{\frac{p(\theta_1)p(y|\theta_1)}{p(y)}}{\frac{p(\theta_2)p(y|\theta_2)}{p(y)}} = \frac{p(\theta_1)}{p(\theta_2)} \cdot \frac{p(y|\theta_1)}{p(y|\theta_2)} \tag{2.23}$$

What this formulation states is that the prior odds $\frac{p(\theta_1)}{p(\theta_2)}$ can be multiplied by the likelihood ratio $\frac{p(y|\theta_1)}{p(y|\theta_2)}$, in order to mathematically obtain the posterior odds.

A necessary supplement to Bayesian inference in the field of Bayesian statics within machine learning and as a whole, is that of *Gaussian processes*, which will be introduced succeedingly.

2.2.4 Gaussian Distributions and Processes

Following the introduction of Bayesian inference, a subsequent introduction to Gaussian distributions is necessary, as these distributions constitute the basis of Bayesian statistics. Gaussian processes are a crucial concept within the classical Bayesian framework, that offer reliable measures of predictive uncertainty (Rasmussen and Williams 2005). Their efficacy has been demonstrated in various domains, including transfer learning (Kandemir 2015), deep learning (Wilson et al. 2016), and active learning (Houlsby et al. 2012). Therefore, in order to fully understand the machine learning techniques relying on Gaussian processes, it is necessary to introduce their underlying theoretical principles. Rasmussen and Williams (2005) provide a thorough theoretical introduction to Gaussian processes within machine learning.

Gaussian processes are extensions of the Gaussian probability distribution, also known as the *normal probability distribution*, that offer a useful tool for statistical modeling by providing a measure of confidence when predicting functions. This feature enables the incorporation of prior knowledge when making predictions in

regression, classification, and clustering tasks (Kim and J. Lee 2007; Kapoor et al. 2010). For any given set of data points, there exist multiple functions that could fit said data, potentially infinite in number. Gaussian processes account for this by assigning probabilities to each of these functions, in order to represent the most probable generalization of the data, using the mean of the probability distributions. The multivariate version of the Gaussian distribution is particularly relevant due to its ability to accurately model complex systems, making it a valuable tool for handling real-world phenomena.

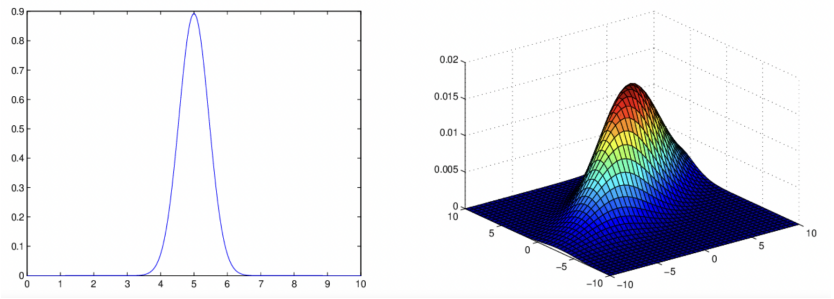


Figure 2.2.4: Illustration of a uni-variate Gaussian density over a single variable and a multivariate Gaussian density over two variables, in the left and right plot, respectively, presented by Görtler, Kehlbeck and Deussen (2019).

Generally, the *multivariate Gaussian distribution*, also known as the *joint normal distribution*, is based on the definition of "*a random variable being k-variate normally distributed if every linear combination of its k components have a univariate normal distribution*" (Shi 2019). This entails the mathematical property of $X = (X_1, \ldots, X_k)^T$ being multivariate Gaussian distributed, assuming $Y = a_1X_1 + a_2X_2 + \ldots + a_kX_k$ is normally distributed, for any constant real vector $a \in \mathcal{R}^k$.

Following the notation presented by Görtler, Kehlbeck and Deussen (2019), based on the underlying assumptions as presented, Equation (2.24) can be used to state a vector-valued random variable X with a multivariate Gaussian distribution, in terms of its mean vector μ and corresponding covariance matrix Σ, essentially meaning that the vector X follows a normal distribution.

$$X = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_n \end{bmatrix} \sim \mathcal{N}(\mu, \Sigma)\,. \tag{2.24}$$

The two terms, namely the mean vector μ and covariance matrix Σ, can be viewed as measurements of the distribution's expected value, based on each dimension's mean, and its corresponding shape, determined by the correlation between the independent variables, respectively. The mean vector, as mentioned, is solely dependent on the mean value of each dimension, where as the covariance matrix is obtained through the mathematical property formulated in terms of two independent variables as presented in Equation (2.25).

$$\Sigma = Cov(X_i, X_j) = E[(X_i - \mu_i)(X_j - \mu_j)^T] \tag{2.25}$$

This mathematical relationship is expressed in the matrix in Equation (2.26), where the $i-th$ diagonal of the matrix represents the i-th independent variable's variance, and the correlation between the i-th and j-th variable is represented by the ij-th element of the matrix.

$$\Sigma = \begin{bmatrix} \sigma_1^2 & 0 & \dots & 0 \\ 0 & \sigma_2^2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \sigma p^2 \end{bmatrix} \tag{2.26}$$

A significant benefit of the multivariate Gaussian distribution is the fact that it adheres to the same underlying assumptions as the *Central Limit Theorem*, which expresses that the mean vector X tends toward a multivariate Gaussian distribution for large samples, assuming the elements of said vector $\{X_1, X_2, \dots, X_n\}$ are independent and identically distributed (I.I.D.) variables. Gaussian processes possess another critical trait in regard to uncertainty modeling, which is their ability to function as universal predictors, meaning they are capable of fitting a broad range of nonlinear prediction functions to data, at the same time enabling the calculation of the variance of predictions through closed-form expressions (Tran, Ranganath and Blei 2015). Moreover, Gaussian processes are non-parametric models, and thus do not contain any stochastic model parameters.

A crucial feature of Gaussian distributions that makes them useful within several domains is the fact that they are are closed under so-called *marginalization* and *conditioning*. Essentially, what this means, is that these mathematical operations can be performed on the distributions, with the resulting distribution being a modified Gaussian itself.

Generally, conditioning can be viewed as a process of revising a probability distribution by incorporating novel information. Specifically, this means that the distribution of one variable is updated based on observations of other variables. Within a Bayesian framework based on Gaussian processes, this entails a prior distribution over functions being updated to a posterior distribution where the new observations are incorporated. Due to the fact that uncertainty is modeled as a distribution over possible functions that could fit a dataset within Gaussian statistics, conditioning allows for capturing the posterior distribution over these functions by conditioning a prior distribution on the observed data.

Marginalization is a mathematical operation used in order to obtain the distribution of a subset of variables, by integrating out certain variables from a joint probability distribution (Rasmussen and Williams 2005). Within a Gaussian process, this entails capturing the posterior distribution over the function values considering a set of test points, assuming there already exists a prior distribution over the function itself and the observed data.

By considering two subsets of original random variables, X and Y, obtained in Equation (2.27), the aforementioned operations can be formulated mathematically.

$$P_{X,Y} = \begin{bmatrix} X \\ Y \end{bmatrix} \sim \mathcal{N}(\mu, \Sigma) = \mathcal{N}\left(\begin{bmatrix} \mu_X \\ \mu_Y \end{bmatrix}, \begin{bmatrix} \Sigma_{XX} \Sigma_{XY} \\ \Sigma_{YX} \Sigma_{YY} \end{bmatrix} \right) \tag{2.27}$$

As mentioned, marginalization allows for capturing partial information from multivariate probability distribution, resulting in marginalized probability distributions, which, mathematically, are denoted as

$$\begin{aligned} X &\sim \mathcal{N}(\mu_X, \Sigma_{XX}) \\ Y &\sim \mathcal{N}(\mu_Y, \Sigma_{YY}) \end{aligned} \tag{2.28}$$

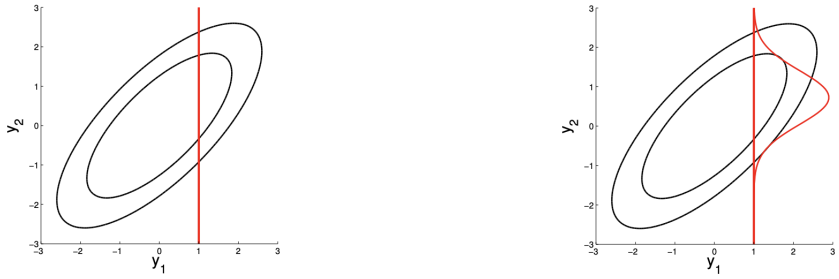
assuming a Gaussian distribution $P(X, Y)$ over vectors of random variables, X and Y. Essentially, what this formulation states is that the partitions, or subsets, are merely dependent on their corresponding entries in the mean and covariance, μ and Σ. Further elaborating on this notion, one can marginalize out a desired random variable from the Gaussian distribution, simply by dropping the variables from μ and Σ. Should the probability density of $X = x$ be desirable, for instance, all possible combinations of outcomes of Y that could lead to this result need to be considered, leading to the formulation stated in Equation (2.29).

$$p_X(x) = \int_y p_{X,Y}(x, y) dy = \int_Y p_{X|Y}(x, y) p_Y(y) dy \tag{2.29}$$

Conditioning, as a means of examining the probability of one variable being dependent on another, is a key aspect of Gaussian processes, as it essentially constitutes the basis for *Bayesian inference*. As mentioned, the operation is closed and therefore produces a modified Gaussian. Conditioning is defined in Equation (2.30), where the new mean solely depends on the conditioned variable, as opposed to the covariance matrix, which does not depend on said variable.

$$\begin{aligned} X|Y &\sim \mathcal{N}(\mu_X + \Sigma_{XY}\Sigma_{YY}^{-1}(Y - \mu_Y), \Sigma_{XX} - \Sigma_{XY}\Sigma_{YY}^{-1}\Sigma_{YX} \\ Y|X &\sim \mathcal{N}(\mu_Y + \Sigma_{YX}\Sigma_{XX}^{-1}(X - \mu_X), \Sigma_{YY} - \Sigma_{YX}\Sigma_{XX}^{-1}\Sigma_{XY} \end{aligned} \tag{2.30}$$

Turner (2016) provides a 2D illustration of the operation of conditioning, in Figure 2.2.5, where the value of one variable in a distribution is fixed, in order to be able to compute the corresponding probability density of the other variable, conditioned on the fixed variable.



(a) A variable y_1 fixed to a singe value, allowing for conditioning to be performed.

(b) Probability density of the variable y_2 computed based on the fixed variable y_1.

Figure 2.2.5: 2D Conditioning example provided by Turner (2016). The variable y_1 is fixed, as shown in (a), which enables the computation of the probability density of y_2 based on the fixed variable, as shown in (b).

The resulting distribution from this operation is a modified Gaussian, with a corresponding mean and covariance, μ and Σ, illustrated in Figure 2.2.6.

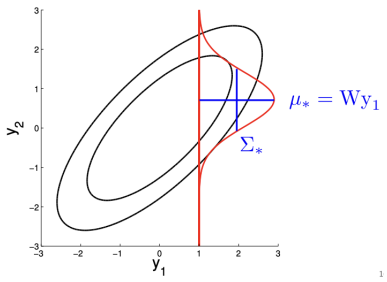


Figure 2.2.6: Modified Gaussian distribution resulting from the conditioning operation, presented by Turner (2016). The resulting probability density following the operation is a modified Gaussian, due to the property $y_2 \in \mathcal{N}(\mu, \Sigma)$.

These theoretical underpinnings make up the core of so-called *Gaussian Processes*, which are characterized by Shi (2019) as *probability distributions over possible functions that fit a set of data-points*. As such, these processes have a wide array of applications within the context of machine learning and uncertainty quantification, especially in terms of regression and classification tasks. Another descriptive feature of Gaussian processes in a machine learning framework is that they are non-parametric, meaning, they do not assume any specific relationship between the input space and desired outputs, but rather are concerned with capturing the predictive uncertainty when mapping from input to output, providing some measure of reliability alongside the inference itself.

The measure of uncertainty related to inferences can be obtained directly, as the Gaussian processes provide an assembly of potential functions that describe a set of observations, through the related uncertainty of the estimated regression function, as illustrated by Hüllermeier and Waegeman (2021) in Figure 2.2.7.

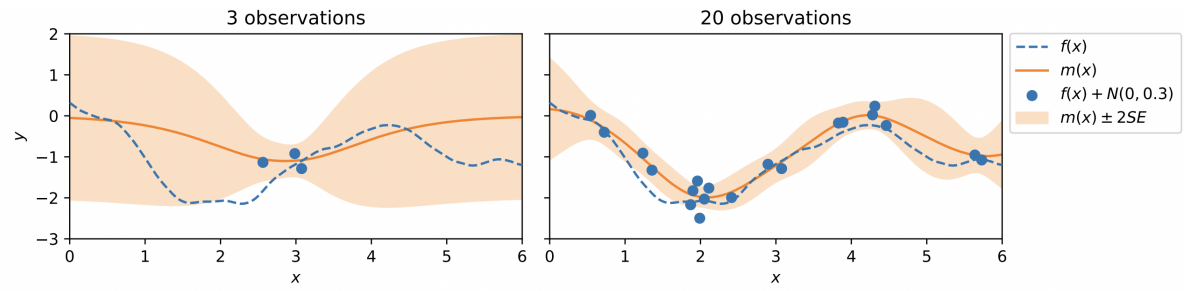


Figure 2.2.7: Hüllermeier and Waegeman (2021) provide an example of a univariate Gaussian process, where the associated uncertainty of the estimated regression function $f(x)$ is represented by the orange band surrounding the function. The uncertainty around the estimated regression function is reduced as new information, in the form of data-points, is introduced. $M(x)$ depicts the ideal function describing the data.

Following the introduction of Bayesian inference and Gaussian processes, the theoretical foundations of specific methodologies in the field of uncertainty quantification can be presented, in light of this underlying statistical framework.

CHAPTER

THREE

METHODS

In the following section certain specific methodologies from the field will be presented, to provide an insight into currently available approaches. The methods have been selected on account of a variety of factors, ranging from their prevalence in the existing literature, their efficacy and proven results, as well as their potential for further improvements and applications. Specifically, *Monte Carlo Dropout*, *Deep Ensembles*, *Prior Networks*, *Evidential Deep Learning* and *Temperature Scaling* will be introduced. A comprehensive discussion regarding the optimal areas of use, advantages and disadvantages of the different methodologies and how they compare will be presented in Section 5.

3.1 Monte Carlo Dropout

Monte Carlo Dropout is a method for capturing model uncertainty first introduced by Gal and Ghahramani (2016). The method is based on the realization that employing dropout before every weight layer of a neural network can be interpreted as a Bayesian approximation of the Gaussian process, thus allowing for information regarding the uncertainty surrounding the inferences to be obtained. As mentioned in Section 2.2.1, dropout is traditionally used as a regularization technique designed to combat overfitting of model parameters to training data, whereas, in this case, the inherent functionality is exploited in order to generate information regarding the predictive uncertainty of the model.

A significant benefit of the method is the fact that existing model architectures do not need to be altered in order to include the desired functionality, as the method merely relies on utilizing dropout in the neural network's layers during the inference of the model, rather than during the training procedure. This allows pre-existing, trained models to adopt the method, reducing the need for potentially complex training and developmental procedures. By performing dropout during inference multiple distinct predictions are produced for each input sample, removing any potential determinism from the model, meaning distinct outputs will be generated for the same sample for each forward pass through the network. The uncertainty and corresponding variance of generated predictions can then be

evaluated, by inferring a predictive distribution $p(y|x, \mathcal{D})$, assuming an input x, target y and N training samples $\mathcal{D} = (x_i, y_i)_{i=1}^{N}$. This foundation essentially constitutes the basis of Monte Carlo Dropout.

The following framework for representing the method notationally was developed by Gal and Ghahramani (2016), with the intention of serving as a formalized structure of Monte Carlo Dropout. Based on said framework, Davis, Zhu and Oldfather (2020) provide some useful formulations, which serve as a useful starting point that will be introduced initially. A distinct sample can be drawn from the approximated parametric posterior distribution $\theta_t \sim q(\theta|\mathcal{D})$, for each dropout configuration θ_t. By doing so, a feasible approximation can be used to represent the predictive distribution, as denoted in Equation (3.1). This approximation follows directly from Equation (3.2), which states the true predictive distribution for variational inference.

$$p(y|x) \sim \frac{1}{T} \Sigma_{t=1}^{T} p(y|x, \theta_t) \text{ such that } \theta_t \sim q(\theta, \mathcal{D}) \tag{3.1}$$

Equation (3.1) formalizes an approximation of the true predictive distribution for variational inference, derived from Equation (3.2). This approximation is necessary as computing the true distribution is not feasible in a deep learning framework.

$$p(y|x) \sim \int_{\Omega} \underbrace{p(y|x, \theta)}_{likelihood} \underbrace{q(\theta, \mathcal{D})}_{posterior} d\theta \tag{3.2}$$

Assuming the likelihood to be Gaussian distributed, the mean $f(x, \theta)$ and variance $s^2(x, \theta)$ can be obtained from simulations of the network and subsequently used to specify the Gaussian function, denoted as $\mathcal{N}$ in Equation (3.3)

$$p(y|x, \theta) = \mathcal{N}(\underbrace{f(x, \theta)}_{mean}, \underbrace{s^2(x, \theta)}_{variance}) \tag{3.3}$$

Following these formulations, the framework presented by Gal and Ghahramani (2016) can be further explored. Considering an $L-$layered model with loss function $E(\cdot, \cdot)$ that generates an output $\hat{y}$ where the true output is y_i for every instance sample x_i assuming $1 \leq i \leq N$, the L_2 cost function can be formulated as

$$L_{dropout} = \frac{1}{N} \sum_{i=1}^{N} E(y_i, \hat{y}) + \lambda \sum_{i=1}^{L} (||W_i||_2^2 + ||b_i||_2^2) \tag{3.4}$$

Here, the weight matrices are denoted as W_l, with corresponding dimensions $K \times K_{i-1}$, and the bias vectors with dimensions K_i for the $i-th$ layer as b_i. λ refers to the weight decay. By exploiting the aforementioned structures, the uncertainty of a neural network can be captured, through the approximate predictive distribution for an instance sample x with the corresponding set of random variables $w = \{W_i\}_{i=1}^{L}$, stated as

$$q(y|x) = \int p(y|x, w) q(w) dw \tag{3.5}$$

where $y \in \mathcal{R}^{\mathcal{N}}$, assuming the output vectors to be $\mathcal{N}$-dimensional for an $L-$layered model. The approximate variational distribution $q(w)$ can be evaluated analytically, by sampling T sets of realization vectors from the Bernoulli distribution, thus obtaining an estimate defined as

$$E_{q(y|x)}\,(y) \approx \frac{1}{T}\,\Sigma_{t=1}^{T}\hat{y}(x, W_1^t, \ldots, W_L^t) \tag{3.6}$$

where $\{W_1^t, \ldots, W_L^t\}_{t=1}^T$ is obtained from the Bernoulli distribution $\{z_1^t, \ldots z_L^t\}_{t=1}^T$ assuming $z_i^t = [z_{i,j}^t]_{j=1}^{K_i}$. The estimate in Equation (3.6) essentially represents averaging T stochastic forward passes through a neural network, and constitutes the foundation of the Monte Carlo Dropout method, through which uncertainty can be quantified. The optimal number of forward passes through the network could be evaluated analytically, although, generally, a range of 30-100 is considered an appropriate range (Gal and Ghahramani 2016). By including the weight decay λ and prior length scale l, the model precision τ can be obtained through the following property, as

$$\tau = \frac{pl^2}{2N\lambda} \tag{3.7}$$

Finally, the predictive log-likelihood can be estimated, which provides a measure of the model's performance in terms of fitting the mean and uncertainty. Provided a dataset X and Y, possible output values y^* based on a new input x^* can be obtained through the predictive probability stated as

$$log\; p(y^*|x^*, X, Y) \sim log\;\left(\frac{1}{T}\Sigma_{t=1}^{T}p(y^*|x^*, w_t)\right) \text{where}\;\; w_t \sim q(w) \tag{3.8}$$

For regression tasks, this log-likelihood is formulated in Equation (3.9). Here, τ refers to the model precision, as obtained through Equation (3.7).

$$\begin{aligned} log\; p(y^*|x^*, X, Y) \sim\; & LogSumExp\,(-\frac{1}{2}\tau||y - \hat{y}_t||^2) \\ & -log\,(T) - \frac{log\,(2\pi)}{2} - \frac{log\,(\tau^{-1})}{2} \end{aligned} \tag{3.9}$$

A visualization of the intended architecture of Monte Carlo Dropout is provided by Davis, Zhu and Oldfather (2020), in Figure 3.1.1. The neural network is altered during each forward pass by distinct dropout configurations. By doing so, several forward passes through slightly distinct model structures can be performed to obtain a predictive distribution $p(f(x, \theta))$ over the mean.

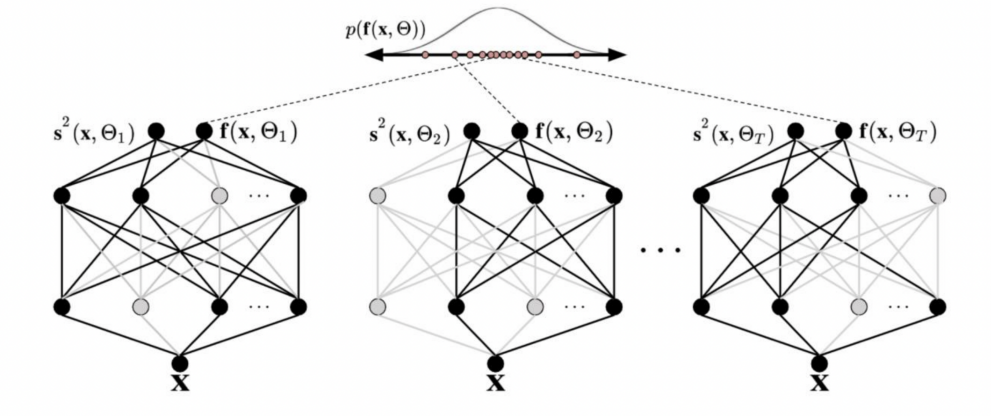


Figure 3.1.1: Example structure of Monte Carlo Dropout in a deep neural network, presented by Davis, Zhu and Oldfather (2020). Several distinct variations of the same model can be seen, each with its own corresponding dropout configuration. By combining the forward passes through the model variations a predictive mean $p(f(x, \theta))$ can be generated, as shown in the generated distribution above the networks.

The fact that uncertainty estimates can be obtained through stochastic forward passes facilitates, as mentioned, the reuse and implementation of pre-existing models, as the method does not require invasive adjustments or redevelopment of the prior model structure. Also, a major advantage stems from the fact that the stochastic forward passes can be performed concurrently, meaning the running time is identical to that of traditional dropout. Despite its simplicity of implementation, the method has performed well within various applications and fields, obtaining state-of-the-art results within a range tasks, some of which are presented in Table 3.1.1, to provide an overview of some existing applications.

Study	Method	Application
Leibig et al. (2017)	CNN	Diabetic Retinophaty
Choi et al. (2017)	Mixture Density Network	Regression
Jungo, McKinley et al. (2018)	Full-Res ResNet	Brain Tumor Segmentation
Jungo, Meier et al. (2018)	FCN	Brain Tumor Segmentation
Vandal et al. (2018)	Variational LSTM	Predicting Flight Delays
DeVries and Taylor (2018)	CNN	Medical Image Segmentation
Tousignant et al. (2019)	CNN	MRI Images
Norouzi et al. (2019)	FCN	MRI Image Segmentation
Roy et al. (2018)	Bayesian FCN	MRI Segmentation
Filos et al. (2019)	CNN	Diabetic Retinophaty
Harper and Southern (2022)	RNN and CNN	Emotion Prediction

Table 3.1.1: Overview of studies on the effectiveness of Monte Carlo Dropout within a range of applications.

Experiments relating to Monte Carlo Dropout and a thorough discussion based on the introduced foundations will be presented in Sections 4 and 5, respectively.

3.2 Deep Ensembles

Following the realization that dropout can be used for uncertainty estimation, research was performed to investigate whether the same principle could be applied within ensemble methods. Subsequently, *Deep Ensembles*, first presented by Lakshminarayanan, Pritzel and Blundell (2016), were introduced as an alternative to Bayesian methods for capturing model uncertainty. The method has yielded state-of-the-art performances across a variety of machine learning tasks (Krizhevsky, Sutskever and G. E. Hinton 2012; Mikolov et al. 2013; Zhou and Troyanskaya 2015; Yann LeCun, Bengiuo and G. Hinton 2015), and has been shown to outperform Monte Carlo Dropout in capturing out-of-distribution samples (Ovadia et al. 2019; Fort, Hu and Lakshminarayanan 2019), in addition to providing high-quality uncertainty estimates.

Deep Ensembles are built on an underlying theoretical principle analogous to that of the French mathematician Nicolas de Condorcet's jury theorem, which states that, generally, assuming the chance of each distinct juror's decision or verdict to be correct to be greater than 50%, the corresponding probability of the jury as a whole reaching a correct verdict tends toward 100% as the number of jurors increases, given that the decision made by each juror is independent of the others (Estlund 1994). Stated formally, Condorcet's theorem states that during a dichotomous choice, if the competence of each individual comprising a collective is greater than 0.5, then, under the majority rule, the competence of the collective decision comprised of said individuals approaches 1, as either the number of individuals or the competence of the individuals increases.

Generally speaking, ensemble methods can be viewed as learning algorithms that produce inferences by combining the inferences made by underlying models, often referred to as *weak learners*. Essentially, Deep Ensembles is an ensemble method where the ensemble consists of deep neural networks. The inferences produced by the underlying models are usually combined by *weighted* or *unweighted* voting, which allows for certain models to be assigned higher importance or trust than others when producing a final decision. Most of the time these ensembles provide better results than the individual models that they are comprised of, as per Dietterich (2000), and by exploiting this general structure, predictive uncertainty estimates can be obtained. An example of a Deep Ensemble structure is provided by Ashukha et al. (2020), in Figure 3.2.1.

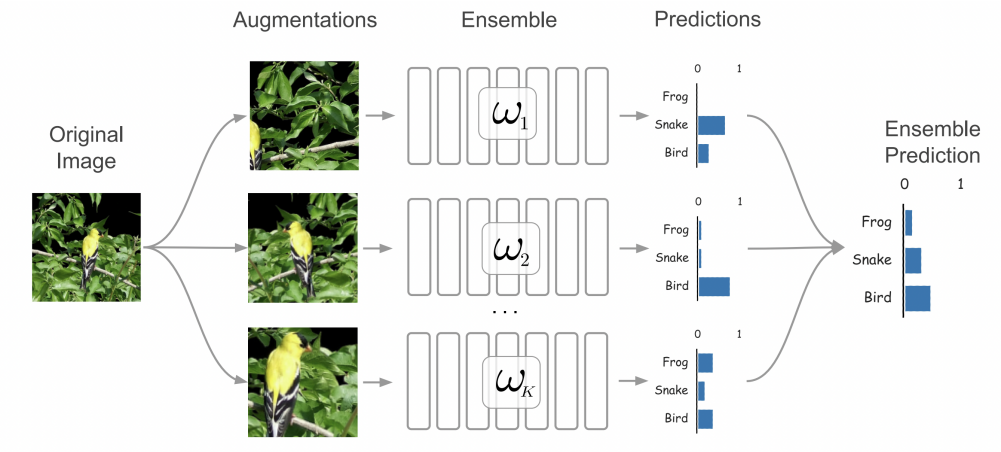


Figure 3.2.1: Example architecture of a Deep Ensemble provided by Ashukha et al. (2020). Several networks, that together constitute an ensemble, are introduced to distinct input samples, which, in this case, are augmentations of the same image. Each model then generates a prediction, which, in turn, is combined with the other generated predictions, to make one final inference. As can be seen in the figure, each model in the ensemble does not necessarily generate the correct prediction, whereas the final averaged prediction output by the ensemble is correct. In this fashion ensembles tend to account for possible misclassifications.

As a general rule, two steps are included in ensembling machine learning models, namely, training the different models and combining the results. As a means of obtaining variations across the models in the ensemble, the training regimes and data, feature subsets and model architectures can be altered. Generating the final inference of the ensemble can be done in multiple ways, some of which include *averaging*, which simply averages the predictions made by each distinct model, *model stacking*, which trains new models on top of the predictions, or one can design custom voting rules, tailored for any desired purpose. The example shown in Figure 3.2.1 uses the averaging method, where the inferences from each model in the ensemble are combined and averaged.

Lakshminarayanan, Pritzel and Blundell (2016) propose a general framework for setting up a Deep Ensemble comprised of three steps, namely

1) *Use a proper scoring rule as the training criterion*

2) *Smooth the predictive distributions by use of adversarial training*

3) *Train the ensemble*

A mathematical framework is needed to facilitate the implementation of the aforementioned steps within a machine learning system, which is also presented by Lakshminarayanan, Pritzel and Blundell (2016). Assuming a dataset $\mathcal{D} = \{x_n, y_n\}_{n=1}^{N}$ comprised of N I.I.D. training samples where $x \in \mathcal{R}^D$ represents the $D-$dimensional features, the probabilistic predictive distribution $p_\theta(y|x)$ with model parameters θ can be modeled by use of a neural network. Note that for regression tasks, $y \in \mathcal{R}$ is assumed, whereas for classification tasks, $y \in \{1, \ldots, K\}$

is assumed, referring to K distinct classes. For an ensemble consisting of M models, $\{\theta_m\}_{m=1}^{M}$ represents the model parameters of the ensemble.

As mentioned in the first of the three steps, the training criterion should be a *scoring rule*, which, essentially, is a function $S(p_\theta, (y, x))$ designed to assign a score to a predictive distribution judged on the perceived quality of the predictions relative to an event $y|x \sim q(y|x)$, where $q(y, x)$ states the true distribution on (y, x)- tuples. As such, the scoring rule is mathematically defined as

$$S(p_\theta, q) = \int q(y, x)\, S(p_\theta, (y, x))\, dydx \tag{3.10}$$

To ensure that this scoring rule to is *proper*, it must assume the strict property stated in Equation (3.11), given that for every p_θ and q, the property $p_\theta(y|x) = q(y|x)$ holds true.

$$S(p_\theta, q) \leq S(q, q) \tag{3.11}$$

Following this foundation, a neural network can be trained to calibrate the predictive uncertainty by minimizing the loss, as given in Equation (3.12). By investigating these properties, it becomes evident that several loss functions designed for neural networks are in fact proper scoring rules, one of which being the softmax cross entropy loss, which makes them ideal for use within Deep Ensembles.

$$\mathcal{L}(\theta) = -S(p_\theta, q) \tag{3.12}$$

In regression tasks a single output value is given, whilst the mean-squared-error (MSE) is minimized to optimize the parameters. The MSE is not equipped to capture predictive uncertainty however, which needs to be accounted for. Based on Nix and Weigend (1994)'s work, by utilizing a neural network that outputs the predicted mean $\mu(x)$ and corresponding variance $\sigma^2(x) > 0$ in the final layer, the negative log-likelihood criterion, as given in Equation (3.13), can be minimized, by treating the observed value as an instance from a heteroscedastic Gaussian distribution.

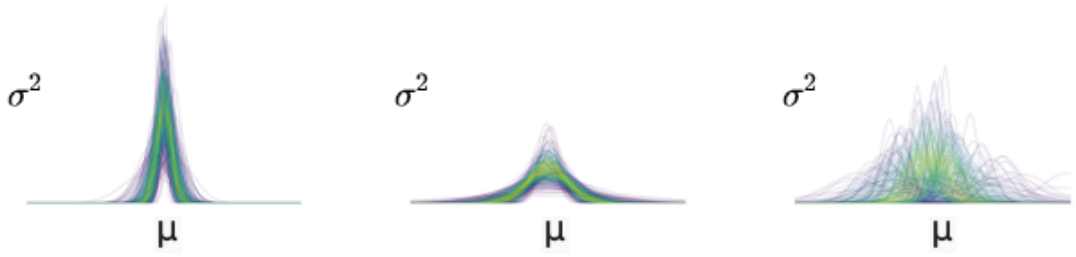
$$-log\, p_\theta(y_n|x_n) = \frac{log\, \sigma_\theta^2(x)}{2} + \frac{(y - \mu_\theta(x))^2}{2\sigma_\theta^2(x)} + C \tag{3.13}$$

The final term C in the above equation refers to a constant. The team found that this approach performed well in their experiments, and that is serves well as a foundation for implementing Deep Ensembles concerned with capturing uncertainty.

Normal distribution samples drawn from a Normal-Wishart distribution, which are higher-order distributions over the parameters of multivariate normal distributions, can be used to visualize the desired behavior of regression models in an ensemble, as presented by Malinin, Chervontsev et al. (2020) in Figure 3.2.2. The drawn samples can be used to detail the optimal behavior of each model, in regard to modeling *low uncertainty*, which essentially represents high confidence, *data uncertainty* and *knowledge uncertainty*, otherwise referred to as aleatoric and epistemic uncertainty. The figure details this by use of the normal distribution

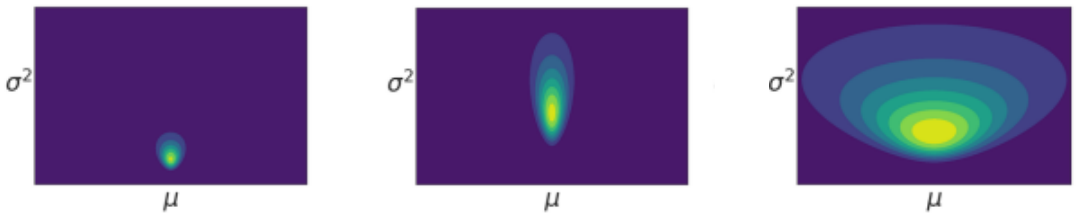
samples, drawn from the corresponding Normal-Wishart distribution, as shown in Figure 3.2.2 and 3.2.3. Both figures represent corresponding scenarios of *low uncertainty*, *data uncertainty* and *knowledge uncertainty*, in the left to right column, respectively.

Ideally, the ensemble should produce a widespread distribution, in terms of its mean and variance, should the observed sample be out-of-distribution, as shown in the rightmost column in Figure 3.2.2. As opposed to this, when encountering in-distribution samples, the desired behavior of the ensemble is for it to be consistent in terms of representing *low uncertainty* and *data uncertainty*, as seen in the left and middle column of the figure. As mentioned, the corresponding Normal-Wishart distributions of the described scenarios are shown in Figure 3.2.3.



(a) Low Uncertainty **(b)** Data Uncertainty **(c)** Knowledge Uncertainty

Figure 3.2.2: Optimal behavior of regression models in an ensemble, presented in terms of the mean μ along the x-axis, and corresponding variance σ^2 along the y-axis, of normal distribution samples drawn from optimal Normal-Wishart distributions, which are shown in Figure 3.2.3. The columns represent three distinct scenarios, namely, *low uncertainty*, *data uncertainty* and *knowledge uncertainty*, from left to right, respectively. The figures are developed by Malinin, Chervontsev et al. (2020).



(a) Low Uncertainty **(b)** Data Uncertainty **(c)** Knowledge Uncertainty

Figure 3.2.3: Visualizations of the Normal-Wishart distributions from which the normal distribution samples in Figure 3.2.2 were drawn, presented in terms of the mean, μ, and corresponding variance, σ^2. Once again, the three columns represent the scenarios of *low uncertainty*, *data uncertainty* and *knowledge uncertainty*, from left to right, respectively (Malinin, Chervontsev et al. 2020).

The second step, as presented by Lakshminarayanan, Pritzel and Blundell (2016), consists of utilizing *adversarial training* to smooth the predictive distributions. Adversarial samples, first introduced by Szegedy (2014), and later improved by Goodfellow (2015), refer to samples that are misclassified by neural networks

but are visually indistinguishable from the original sample to a human being, and should not be confused or related with Generative Adversarial Networks. The robustness of the models can be improved by generating such samples by adding perturbations along the direction that is most likely to increase the loss. By including this procedure, not only is the robustness of the model improved, but the predictive distribution can also be smoothed, by increasing the likelihood of the target sample being close to the observed samples (Goodfellow 2015; Nix and Weigend 2015).

The third and final step is to train the ensemble's underlying models. Generally, the optimal number of members in an ensemble can be evaluated analytically, however, Lakshminarayanan, Pritzel and Blundell (2016) and Ovadia et al. (2019) suggest that five members tends to be a viable starting point. Although Deep Ensembles have attained state-of-the-art results in several tasks, one drawback of the method as compared to Monte Carlo Dropout, is the computational need, as both the ensembling and training procedures are potentially computationally demanding. As mentioned, a further reflection regarding the desired usage and potential advantages and disadvantages of the method will be presented in Section 5, in light of the necessary considerations of the field as a whole.

3.3 Prior Networks

Malinin and Gales (2018) present a novel approach for quantifying uncertainty in deep neural networks, namely a class of models named *Prior Networks*. The underlying idea of Prior Networks is based on integrating prior knowledge regarding the problem domain directly into the model architecture, primarily in the form of information about the data being processed by the network. The prior knowledge in question is not limited to this, however, and could encapsulate other sources of information, such as the physical laws specific to the domain in which the data is contained. A notable additional feature of Prior Networks as compared to previous methods is the ability to explicitly capture and model *distributional uncertainty*, the source of which, in this case, is distinct from aleatoric and epistemic uncertainty.

Distributional uncertainty is a common occurrence in real-world applications that generally stems from data shifts, meaning discrepancies between the training and test data (Quiñonero-Candela et al. 2010). Specifically, this means that the model lacks familiarity with the distribution of the test data, having solely observed the distribution of the available training data. Differentiating this type of uncertainty from other sources is useful in determining and engineering an appropriate response, one of which is to update the current data distribution to encapsulate the out-of-distribution samples.

The underlying theoretical framework of the methodology will be presented in order to provide an understanding of the workings of the method, before highlighting the differences and similarities to other available methods. A brief introduction to the traditional approach to uncertainty estimation in a similar framework

should be presented as to provide the basis of the extended functionalities of the Prior Networks. Using Malinin and Gales (2018)'s notation, generally, the traditional approach for capturing predictive uncertainty in a model is formulated in Equation (3.14), considering a distribution $p(x, y)$ over input features and corresponding labels.

$$P(y|x, \mathcal{D}) = \int \underbrace{P(y|x, \theta)}_{Data} \; \underbrace{p(\theta|\mathcal{D})}_{Model} \, d\theta \tag{3.14}$$

This formulation states that in a Bayesian framework, given a finite dataset $\mathcal{D} = \{x_j, y_j\}_{j=1}^{N} \sim p(x, y)$ for training, the predictive uncertainty of classification models is given as $P(y|x, \mathcal{D})$ for an input sample x. In this framework, the predictive uncertainty of the model stems from a combination of aleatoric and epistemic uncertainty, estimated as the posterior distribution over class labels with corresponding model parameters θ, and the posterior distribution over said parameters given the training data, respectively.

Obtaining the true posterior $p(\theta|\mathcal{D})$ is not necessarily feasible (Kingma, Salimans and Welling 2015), and therefore, an explicit or implicit variational approximation, $q(\theta)$, is usually employed in its stead, meaning $p(\theta|\mathcal{D}) \approx q(\theta)$ (Blundell et al. 2015). The integral, as given in Equation (3.14), oftentimes needs to be approximated by sampling, as it may be computationally infeasible to calculate. Some ways of sampling include, but are not limited to, *Monte Carlo Dropout* (Gal 2016), *Langevin Dynamics* (Welling and Teh 2011) or *Explicit Ensembling* (Lakshminarayanan, Pritzel and Blundell 2016). A formalization of a sampling procedure is provided in Equation (3.15).

$$P(y|x, \mathcal{D}) \approx \frac{1}{M} \Sigma_{i=1}^{M} \, P(y|x, \theta^{(i)}), \; \theta^{(i)} \sim q(\theta) \tag{3.15}$$

When considering an ensemble $\{P(y|x, \theta^{(i)})\}_{i=1}^{M}$, each member, formalized as $P(y|x, \theta^{(i)})$, can be viewed as a categorical distribution μ over labels conditioned on the input sample x. By use of a simplex each of these aforementioned members can be visualized, as a collection of distinct points, which, induced via the posterior distribution over the model parameters, are equivalent to samples of categorical distributions from an implicit conditional distribution. Simplex visualizations are statistical concepts that can be used to illustrate the behavior of Prior Networks, by associating class labels with vertices and representing predictions based on their particular location as points on the simplex. Essentially, the closer a point is to a vertex, the more confident the prediction is.

Malinin and Gales (2018) show this in Figure 3.3.1, as a collection of distinct points representing each member in an ensemble, and the corresponding categorical distribution, in the left and right simplex, respectively.

(a) Ensemble Representation **(b)** Categorical Distribution

Figure 3.3.1: For the same input x each member of an ensemble is visualized as distinct points in the left simplex, alongside the corresponding categorical distribution, in the right simplex (Malinin and Gales 2018).

On this foundation, Malinin and Gales (2018)'s contributions and additions to the presented traditional framework can be introduced. Stated specifically, the goal of Prior Networks, which essentially are DNN's, is to explicitly parameterize a distribution over distributions by training the network to emulate the behavior of traditional implicit Bayesian distributions. The desired behavior of such a model in different scenarios is visualized in Figure 3.3.2, presented by Malinin and Gales (2018)

Ideally, the neural network should produce a sharp distribution in one of the corners of a simplex, as shown in Figure 3.3.2c, to signal confidence in its prediction. Conversely, in the case of an out-of-distribution sample, a flat distribution should be present on the simplex, as is the case in the leftmost simplex in Figure 3.3.2a. For samples located in a region with high data uncertainty, a sharp distribution should be present in the center of the simplex, indicating that the model is confidently predicting a flat categorical distribution over class labels, as seen in Figure 3.3.2b.

(a) Out-of-distribution sample **(b)** High Aleatoric Uncertainty **(c)** High Prediction Confidence

Figure 3.3.2: Three simplices visualizing the desired behavior of the distribution over distributions parameterized by the Prior Network , in terms of *OOD-samples*, *high aleatoric uncertainty* and *high prediction confidence* (Malinin and Gales 2018).

As mentioned, one of the additional functionalities implemented by Malinin and Gales (2018) is the ability to separate *distributional uncertainty* as a distinguished source of uncertainty by including the term $p(\mu|x, \theta)$ to Equation (3.14),

resulting in the formulation as stated in Equation (3.16). Traditionally, distributional uncertainty has been contained within the model uncertainty, as opposed to being viewed as a separate source. Malinin and Gales (2018), however, claim the distinction to be viable and useful, to identify a mismatch between the training and test data's domains. As such, Equation (3.16) states the new structure, which has previously been experimented with and introduced by Blei, Ng and Jordan (2003). Note that in the following formulations, μ refers to the categorical distribution over class labels y conditioned on the input x, sampled from $q(\theta)$.

$$P(y|x,\mathcal{D}) = \int\int \underbrace{p(y|\mu)}_{Data} \; \underbrace{p(\mu|x,\theta)}_{Distribution} \; \underbrace{p(\theta|\mathcal{D})}_{Model} \; d\mu d\theta \tag{3.16}$$

The presented formulations can be used to evaluate the performance of a Prior Network by deriving certain standardized evaluation metrics, all of which have been marginalized from Equation (3.16). *Max probability*, which measures prediction confidence, and *entropy*, which represents the uncertainty related to the predictive distribution, are standardized evaluation metrics within machine learning frameworks, which, combined, can be viewed as an encapsulation of the total uncertainty of a prediction (Tom M. Mitchell 1997; Gal 2016; Hendrycks and Gimpel 2018).

Equation (3.17) states the relevant measure of *max probability* related to a specific class c, marginalized from Equation (3.16), whereas the measure of *entropy* $\mathcal{H}()$, similarly, is stated in Equation (3.18). Combined, these measures represent the total uncertainty of predictions.

$$\mathcal{P} = \max_c \; P(y|x;\mathcal{D}) \tag{3.17}$$

$$\mathcal{H}[P(y|x;\mathcal{D})] = -\Sigma_{c=1}^{K} \; ln(P(y|x,\mathcal{D})) \tag{3.18}$$

Another standardized measure, *Mutual Information* (MI), can be applied to implicitly capture model uncertainty. MI can be viewed as a measurement of mutual dependence between two random variables (Witten, Frank and Hall 2016), which, in this framework, refers to labels y and model parameters θ. By employing this measure it is possible to capture the reduction in uncertainty of a variable, assuming the value of the corresponding variable is known, resulting in the spread of the assembly $P(y|x,\theta^{(i)})_{i=1}^{M}$ being captured. This spread can be interpreted as model uncertainty, implicitly. Equation (3.19) defines the MI measure $\mathcal{I}()$ by marginalizing out μ from Equation (3.16).

$$\underbrace{\mathcal{I}[y,\theta|x,\mathcal{D}]}_{Model\,Uncertainty} = \underbrace{\mathcal{H}[\mathcal{E}_{p(\theta|\mathcal{D})}\,[P(y|x,\theta)]]}_{Total\,Uncertainty} - \underbrace{\mathcal{E}_{p(\theta|\mathcal{D})}\,[\mathcal{H}[P(y|x,\theta)]]}_{Expected\,Data\,Uncertainty} \tag{3.19}$$

This formulation is derived by Depeweg et al. (2018), based on the assumption that the model uncertainty can be obtained by subtracting the expected data

uncertainty from the measure of total uncertainty, captured by the entropy of each member contained in the ensemble. By marginalizing out θ from Equation (3.16) instead, the MI between labels y and μ can be obtained, which, essentially, is an explicit measure of the spread due to the distributional uncertainty rather than the implicit measure in Equation (3.19), as it mimics the behavior of the MI between y and θ.

$$\underbrace{\mathcal{I}[y, \mu | x; \mathcal{D}]}_{Distributional\,Uncertainty} = \underbrace{\mathcal{H}[\mathcal{E}_{p(\mu|x;\mathcal{D})}[P(y|\mu)]]}_{Total\,Uncertainty} - \underbrace{\mathcal{E}_{p(\mu|x;\mathcal{D})}\left[\mathcal{H}[P(y|\mu)]\right]}_{Expected\,Data\,Uncertainty} \tag{3.20}$$

Another measure of uncertainty, *Differential Entropy*, provided in Equation (3.21) can also be used to evaluate a Prior Network, capturing distributional uncertainty by recognizing when the distribution is flat and the corresponding spread of samples is the greatest (Blahut 2002; Pichler et al. 2022).

$$\mathcal{H}[p(\mu|x;\mathcal{D})] = -\int_{S^{K-1}} p(\mu|x;\mathcal{D})\, ln(p)(\mu|x;\mathcal{D})\, d\mu \tag{3.21}$$

In subsequent work Malinin, Chervontsev et al. (2020) extend this framework to encapsulate regression models, obtaining measures of *epistemic, total* and *data uncertainty*. Instead of yielding point-estimate predictions, the distribution $p(y|x,\theta)$ over the target $y \in \mathcal{R}^K$ is parameterized by a probabilistic regression model, resulting in a normal distribution with mean μ and a positive-definite symmetric precision matrix Λ, formalized in Equation (3.22).

$$p(y|x,\theta) = \mathcal{N}(y|\mu,\Lambda) \quad \text{where} \quad \{\mu,\Lambda\} = f(x;\theta) \tag{3.22}$$

Then, to obtain the uncertainty measures, the MI between labels y and the parameters of the output distribution $\{\mu,\Lambda\}$ is evaluated. These measures are stated in Equation (3.23).

$$\underbrace{\mathcal{I}[y, \{\mu, \Lambda\}]}_{Epistemic\,Uncertainty} = \underbrace{\mathcal{H}[\mathcal{E}_{p(\mu,\Lambda|x,\theta}[P(y|\mu,\lambda)]]}_{Total\,Uncertainty} - \underbrace{\mathcal{E}_{p(\mu,\Lambda|x,\theta)}\left[\mathcal{H}[P(y|\mu,\Lambda)]\right]}_{Expected\,Data\,Uncertainty} \tag{3.23}$$

Malinin and Gales (2018) proved the viability of the method by performing well in terms of capturing distributional uncertainty on certain tasks using the *MNIST* and *CIFAR-10* benchmark datasets. Once again, advantages and disadvantages as compared to the other presented methods will be presented in the discussion in Section 5.

3.4 Evidential Deep Learning

Evidential Deep Learning refers to a related class of models that have appeared concurrently to Prior Networks that are structurally similar, but mainly differ in regards to the training regimes (Sensoy, Kaplan and Kandemir 2018; Amini et al. 2019). The methodology is based on the notion that learning is an evidence-acquisition process, by combining the *Dempster-Shafer Evidence theory* (Sentz and Ferson 2002) with *Subjective Logic* (Jøsang 2016), aiming to increase the

predictive confidence by means of introducing additional evidence. Specifically, Evidential Deep Learning is concerned with predicting a *Dirichlet distribution* of class probabilities, which is then evaluated as an evidence acquisition process (Bao, Yu and Kong 2021). Estimating the parameters of this distribution enables modeling aleatoric uncertainty as part of the predictive process.

As such, to understand the method's underlying theoretical foundations, the *Dempster-Shafer Evidence Theory* and *Subjective Logic* need to be introduced. The former aims to generalize Bayesian theory to subjective probabilities, as a scheme for expressing uncertainty inside an Evidential Deep Learning framework (Heendeni et al. 2008). This is accomplished by considering a set of propositions and subsequently assigning an interval of the degree of belief in which the set must lie. The theory is also equipped for handling out-of-distribution samples, as the probability theory enables certain beliefs to be unassigned to any of the potential candidates as a means of representing ignorance or insufficient knowledge.

As opposed to traditional Bayesian methods, which mostly aim to indirectly infer prediction uncertainty through weight uncertainties, Sensoy, Kaplan and Kandemir (2018) aimed to explicitly model this by use of Subjective Logic, which formalizes the Dempster-Shafer belief statements as a Dirichlet distribution. This specific distribution is commonly used as a prior distribution in Bayesian statistics, following Jøsang (2016)'s structure. Combining the two, a framework can be concretized , which will be presented in this section. The notation and derivation for said framework is presented by Sensoy, Kaplan and Kandemir (2018). Note that this primary implementation is based on classification models, whereas extended functionality to regression models will be introduced subsequently.

Considering K mutually exclusive class labels, hereby referred to as *singletons*, with a corresponding belief mass b_k assigned to each $k = \{1, \ldots, K\}$, as well as an overall uncertainty mass u, the identity as described in Equation (3.24) applies, where the total sum equals one. The belief mass is a measure of belief or support assigned to a specific hypothesis, in this case classes for a specific sample, representing the degree to which the available evidence supports the hypothesis.

$$u + \Sigma_{k=1}^{K} b_K = 1 \tag{3.24}$$

This identity applies for $b_k \geq 0$ and $u \geq 0$, where the belief mass b_k for each singleton is calculated using the evidence for said singleton, described by $e_k \geq 0$ for the k^{th} singleton. On this basis, the belief mass b_k and uncertainty mass μ can be defined as

$$b_k = \frac{e_k}{S} \quad , \quad u = \frac{K}{S} \tag{3.25}$$

where $S = \Sigma_{i=1}^{K}(e_i + 1)$. This correlates directly to the behavior of a Prior Network, as a subjective opinion, stated as a belief mass $b = (b_1, \ldots, b_K)$, can be seen as a sample from a Dirichlet distribution. This is done by using a probability density function on a vector p, with corresponding parameters $\alpha = (\alpha_1, \ldots, \alpha_K)$, resulting in the formulation as presented in Equation (3.26). The Dirichlet distribution can then be viewed as a probability density function for possible values of

said vector p.

$$\underbrace{D(p|\alpha)}_{Dirichlet} = \begin{cases} \frac{1}{B(\alpha)} \prod_{i=1}^{K} p_{i=1}^{(\alpha_{i-1})} & \text{for } p \in S_K \\ 0 & \text{otherwise} \end{cases} \tag{3.26}$$

In this formulation, $B()$ refers to the K-dimensional multinomial Beta function (Kotz, Narayanaswamy and Johnson 2000), and S_K refers to the K-dimensional unit simplex, stated as

$$S_K = \left\{ p | \sum_{i=1}^{K} p_i = 1 \text{ and } 0 \leq \{p_1, \ldots, p_K\} \leq 1 \right. \tag{3.27}$$

Equation (3.28) represents the relationship to the Dirichlet distribution, as the expected probability for the $K-th$ class label, considering an opinion, is the mean of the corresponding Dirichlet distribution, calculated as

$$\hat{p}_k = \frac{\alpha_k}{S} \tag{3.28}$$

The team claims that DNN's are inherently equipped to form opinions regarding classification tasks by means of Dirichlet distributions, based on the presented formulations. Specifically, whenever a novel observation regarding a sample is related to an attribute, the Dirichlet parameter can be adjusted in order to adjust the corresponding distribution, essentially using the parameters of the Dirichlet distribution to represent the evidence for each class.

Then, the epistemic uncertainty can be obtained from Equation (3.25), assuming $\alpha_i = \{\alpha_{i1}, \ldots, \alpha_{iK}\}$ to be the Dirichlet distribution's parameters for a sample i, as $(\alpha_{ij} - 1)$ can be used to represent the total evidence estimated by the DNN for the assignment of the sample i to the $j - th$ class. The mean-squared-error (MSE) can be minimized to optimize and train a Deep Evidential Learning model structured as presented, based on a sample X and one-hot labels y_i. The formulation of this loss function is denoted as

$$\mathcal{L}_i(\Theta) = \mathcal{E}_{p_i} \sim D(p_i|\alpha_i)[||y_i - p_i||_2^2] = \Sigma_{j=1}^{K}(y_{ij} - \hat{p}_{ij})^2 + \frac{\hat{p}_{ij}(1 - \hat{p}_{ij})}{(S_i + 1)} \tag{3.29}$$

Amini et al. (2019) further extend the work presented by Sensoy, Kaplan and Kandemir (2018) to encapsulate regression models. The foundations of the framework is modeled on a maximum likelihood perspective, based on maximizing the likelihood of observing a specific set of training data, assuming the targets y_i are I.I.D. and obtained from a Gaussian distribution with mean and variance parameters $\theta = (\mu, \sigma^2)$. The objective is to infer the parameters θ that maximize the likelihood of observing specific targets, defined by $p(y_i|\theta)$. One approach for obtaining the aleatoric uncertainty related to a process modeled by Deep Evidential Learning is to learn these model parameters θ, captured through minimizing the negative log-likelihood loss function, given as

$$\mathcal{L}_i(w) = -log\, p(y_i| \underbrace{\mu, \sigma^2}_{\theta}) = \frac{1}{2}\, log(2\pi\sigma^2) + \frac{(y_i - \mu)^2}{2\sigma^2} \tag{3.30}$$

One drawback of this procedure, however, is the fact that the epistemic uncertainty cannot be obtained concurrently, meaning a specialized approach is necessary. One such approach consists of placing high-order prior distributions over the learned parameters of the observed samples' distribution. Amini et al. (2019) provide a illustration of this structure, in Figure 3.4.1. The figure details the architecture of a Deep Evidential Learning model, where the model learns a continuous target with corresponding measures of aleatoric and epistemic uncertainty, based on inferring the parameters of an *evidential prior distribution*. This prior distribution ideally models higher-order probability distributions over the parameters $\theta = (\mu, \sigma^2)$.

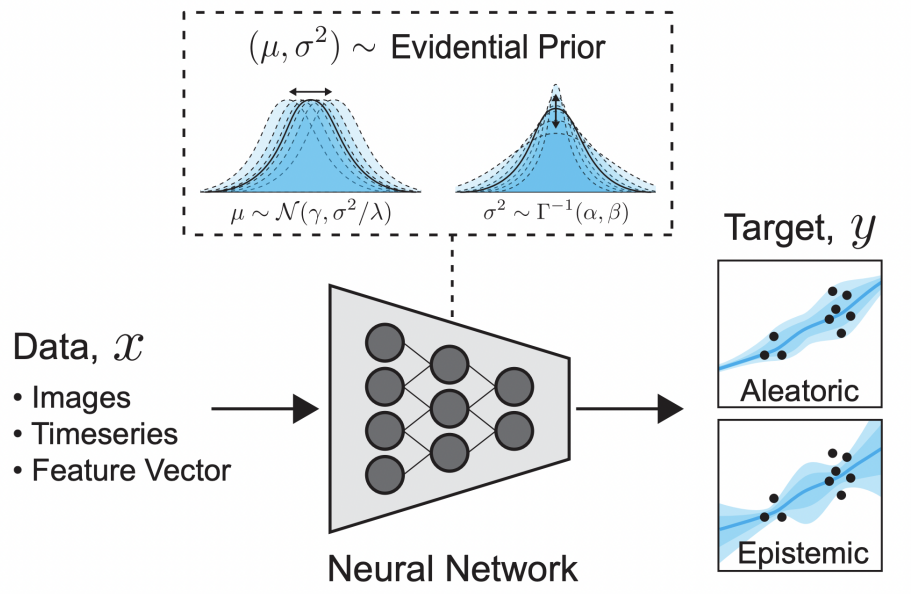


Figure 3.4.1: Amini et al. (2019) visualize the overall structure of Deep Evidential Learning models for input data x. The maximum likelihood optimization approach involves estimating the probability distribution of the data by learning from available information. In contrast, the evidential distribution technique focuses on constructing probability distributions of higher order that encapsulate the probability distribution of the parameters that define the likelihood distribution. These distributions will be further introduced.

The relationship between high order evidential distributions and low order likelihood distributions is illustrated in Figure 3.4.2, this time in terms of the parameters μ and σ^2. The degree of certainty in the model parameters at the higher-order distribution, specifically the variance σ^2 and mean μ, results in the derivation of lower-order distributions over the data, $p(y|\mu, \sigma^2)$.

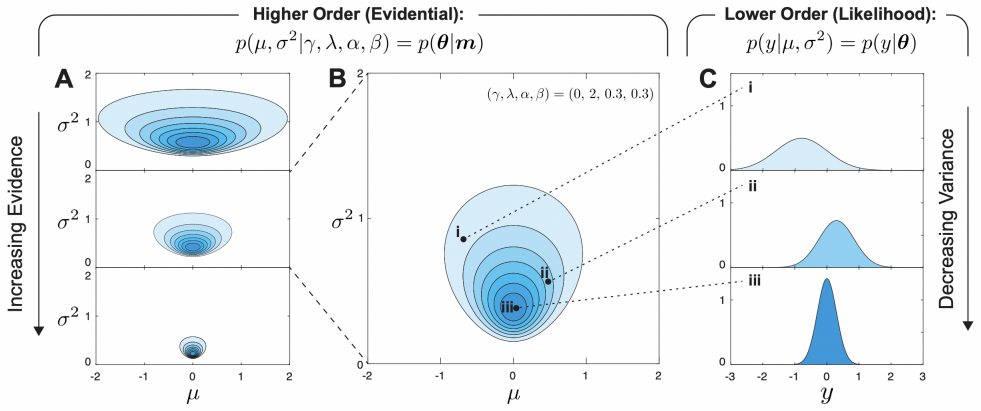


Figure 3.4.2: Illustration of the relationship between high-order evidential distributions and low-order likelihood distributions, in terms of μ and σ^2, presented by Amini et al. (2019). The left quadrant, (A), shows the evidential distribution from which the single realization sampling in (B) is drawn, and the relationship to the corresponding low-order likelihood distributions, in (C). Denser probability mass is indicated by use of darker blue shading.

The evidential prior distributions, as presented in Figures 3.4.1 and 3.4.2, are placed on unknown model parameters $\theta = (\mu, \sigma^2)$ with the intention of probabilistically estimating them. This is done by initially placing a Gaussian prior on the unknown mean, and an Inverse-Gamma prior on the unknown variance, which approximates the Gaussian conjugate prior, referred to as the Normal-Inverse Gaussian (NIG) distribution. Assuming $(y_i, \ldots, y_N) \sim \mathcal{N}(\mu, \sigma^2)$, the formulations in Equation (3.31) can be derived.

$$\mu \sim \mathcal{N}(\gamma, \sigma^2 \lambda^{-1}) \quad and \quad \sigma^2 \sim \Gamma^{-1}(\alpha, \beta) \tag{3.31}$$

In these formulations the Gamma function is represented by $\Gamma()$ and the NIG parameters by γ, λ, α and β. Single instances $\mathcal{N}(\mu_i, \sigma_i^2)$ from the likelihood function can be obtained from the NIG distribution by drawing samples of θ, from which the parameters of the NIG distribution may be used to obtain the uncertainty. Based on this foundation, Amini et al. (2019) present two terms that make up the loss function used to train the Evidential Deep Learning model, as stated in Equation (3.32). The statement is based on maximizing and regularizing the evidence, scaled by a regularization coefficient λ. This formulation is useful as it provides a guideline for training a model based on a similar structure.

$$\mathcal{L}_i(w) = \mathcal{L}_i^{SOS}(w) + \lambda \mathcal{L}_i^R(w) \tag{3.32}$$

The two terms that form the above statement are described in Equations (3.33) and (3.34). $\mathcal{L}_i^{SOS}(w)$ represents the negative log-likelihood, derived from the sum-of-square deviations. After performing evaluations in regards to the efficacy of this function, the approach was deemed desirable, as it provided the most stable performance.

$$\mathcal{L}_i^{SOS}(w) = log\left(\frac{\beta(1+\lambda)}{\lambda} + (\alpha - 1)(y_i - \gamma)^2\right) + log\left(\frac{\Gamma(\alpha - 1)}{\Gamma(\alpha)}\right) \tag{3.33}$$

The conventional practice in variational inference has been to utilize Kullback-Leibler divergence to minimize the disparity between a derived posterior and a prior, thereby ensuring that the training process is properly constrained. However, for NIG distributions characterized by low levels of evidence, such a procedure is deemed ill-defined (Soch and Allefeld 2016), leading to the formulation of a specialized regularizer for evidence, as outlined below.

$$\mathcal{L}_i^R(w) = |y_i - \mathcal{E}[\mu_i] \times \phi = |y_i - \gamma| \times (2\lambda + \alpha) \tag{3.34}$$

The underlying rationale behind utilizing this regularizer is to penalize prediction errors and scale with the total evidence of the inferred posterior. Finally, based on the NIG distribution definitions, as stated in Equation (3.35), the aleatoric and epistemic uncertainty of the system can be obtained, through the mean and variance, as is the standard approach.

$$\underbrace{\mathcal{E}[\sigma^2] = \frac{\beta}{\alpha - 1}}_{Aleatoric\ \ Uncertainty} \quad \text{and} \quad \underbrace{Var[\mu] = \frac{\beta}{(\alpha - 1)\lambda}}_{Epistemic\ \ Uncertainty} \tag{3.35}$$

Similarly, the prediction itself is obtained through the property presented in Equation (3.36).

$$\mathcal{E}[\mu] = \gamma \tag{3.36}$$

The results Sensoy, Kaplan and Kandemir (2018) and Amini et al. (2019) obtained from their experiments suggest that Evidential Deep Learning could hold certain advantages as compared to other traditional methods within the field, concluding that *"the efficiency, scalability, and calibration of our approach could enable the precise and fast uncertainty estimation required for robust NN deployment in safety-critical prediction domains"* (Amini et al. 2019). A comprehensive discussion regarding the advantages and areas of use, as well as potential limitations of the method will be presented in Section 5.

3.5 Temperature Scaling

Temperature Scaling is a post-processing calibration method developed by Guo et al. (2017) intended for a slightly different area of application as compared to the other presented methods. The primary objective of Temperature Scaling is to re-calibrate prediction probabilities in classification models, to address the issue of overconfident prediction values, with the intention of obtaining well-calibrated confidence estimates. However, while the method has been acknowledged as a means of quantifying total predictive uncertainty in the calibrated probabilities (Davis, Zhu and Oldfather 2020), it is not sufficiently equipped to explicitly capture aleatoric and epistemic uncertainty, particularly in terms of out-of-distribution samples (Ovadia et al. 2019).

Generally, *calibrated* confidence predictions refer to probabilities that indicate the extent to which a prediction corresponds to the likelihood of correctness with respect to the ground truth. In spite of its simplicity, the technique of Temperature Scaling has been shown to be remarkably effective at calibrating classification

predictions. The findings suggest that the method can serve as a valuable supplementary feature for any classification model that aims to generate confidence-aware predictions, particularly due to the fact that it can be readily implemented into existing deep learning architectures, removing the need for intrusive or excessive model structure adjustments.

Guo et al. (2017) present a standardized approach for implementing Temperature Scaling as a probabilistic model prediction calibrator for multi-class classification tasks, based on the core theoretical foundations of Jaynes (1957), as well as the introduction of the theory into the field of machine learning by G. Hinton, Vinyals and Dean (2015a).

Essentially, a single scalar parameter $T > 0$, referred to as the *temperature*, constitutes the theoretical basis of the method. This parameter is used to re-scale the logit scores traditionally used to calculate the softmax outputs. Usually, the value of this parameter can be evaluated analytically, using the validation set. As such, the method can be described as a single-parameter version of *Platt Logistic Scaling*, previously introduced by Platt (1999).

Utilizing the notation presented by Guo et al. (2017), assuming a logit vector z_i, the calibrated confidence prediction can be stated as

$$\hat{q}_i = \max_k \sigma_{SM} \left(\frac{z_i}{T}\right)^k \tag{3.37}$$

assuming $K > 2$ classes, where σ_{SM} refers to the softmax function, as previously presented in Equation (2.15). The single scalar value T dictates the behavior of the predicted confidence, as different scenarios arise depending on the value of said temperature, namely when:

$T \to \infty$: As the temperature T approaches infinity, the probability $\hat{q}$, as defined in Equation (3.37), converges towards $\frac{1}{K}$, which can be interpreted as the upper bound of possible uncertainty.

$T > 1$: The act of raising the output entropy of the softmax function effectively results in the moderation of the softmax.

$T = 1$: A temperature of 1 corresponds to the initial probability $\hat{p}_i$.

$T \to 0$: A temperature of 0 leads to the corresponding probability being reduced to a point mass, i.e. $\hat{q}_i = 1$.

The optimal value of T can be evaluated analytically by minimizing the negative log-likelihood on a held-out validation dataset, given a probabilistic model $\hat{\pi}(Y|X)$ with n samples, as stated in Equation (3.38).

$$\mathcal{L} = -\Sigma_{i=1}^{n} log(\hat{\pi}(y_i|x_i)) \tag{3.38}$$

A crucial characteristic of Temperature Scaling is that the technique does not influence the model's overall accuracy, as the temperature parameter T does not

directly impact the softmax function's maximum value. Consequently, the model's class predictions remain unmodified. An illustration of the structure of a neural network employing Temperature Scaling is provided in Figure 3.5.1.

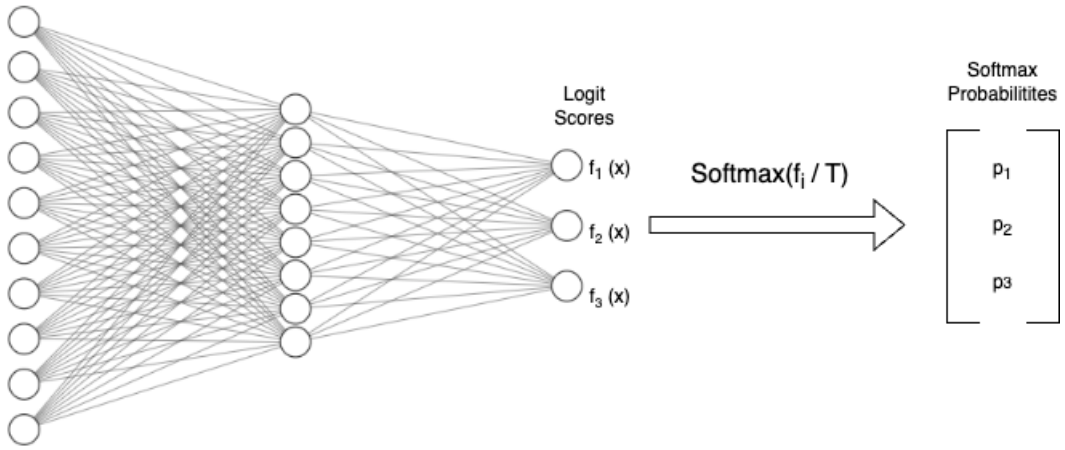


Figure 3.5.1: Example architecture that depicts the structure that emerges following implementation of Temperature Scaling in a neural network. The logit scores produced by the neural network are given as input to the softmax function, along with the temperature T, which is utilized to adjust the scale of the resulting probability scores, as specified. As is evident from the illustration, the method is post-hoc, and thus requires no altercations to the model structure itself.

Standardized evaluation metrics for evaluating the efficacy of the performed calibration exist, one of which will be presented in Section 4.6, alongside a practical implementation of the method. Based on the effectiveness and simplicity of Temperature Scaling as a means of calibrating prediction values, the method should be considered a possible supplement to any classification model concerned with generating uncertainty-aware inferences. Note that due to the fact that the method is based on the re-scaling softmax logits, it is merely applicable in classification contexts and not regression tasks.

CHAPTER

FOUR

PRACTICAL EXPERIMENTS

In the upcoming section practical implementations of Monte Carlo Dropout, Deep Ensembles, and Temperature Scaling will be presented. These specific methods have been selected as the primary focus for the practical experiments due to various reasons, primarily centered around feasibility, which will be elaborated upon in Section 5.

4.1 Softmax Output Compared to Monte Carlo Dropout Probabilities

As discussed in Section 2.2.2, one of the major drawbacks of the traditional use of the softmax activation function is the function's inherent inability to capture overall prediction confidence or uncertainty. As such, a natural starting point with regard to evaluating the effectiveness of Monte Carlo Dropout in mitigating this deficiency is to compare the outputs produced by a traditional neural network with a softmax activation layer, to the prediction probabilities obtained from utilizing Monte Carlo Dropout in the same network.

To explore this feature an experiment based on the LeNet-5 convolutional neural network, originally presented by LeCun et al. (1998), was performed. The model, whose architecture is presented in Figure 4.1.1, was trained on the MNIST dataset introduced by Deng (2012), and subsequently used to classify samples from said dataset.

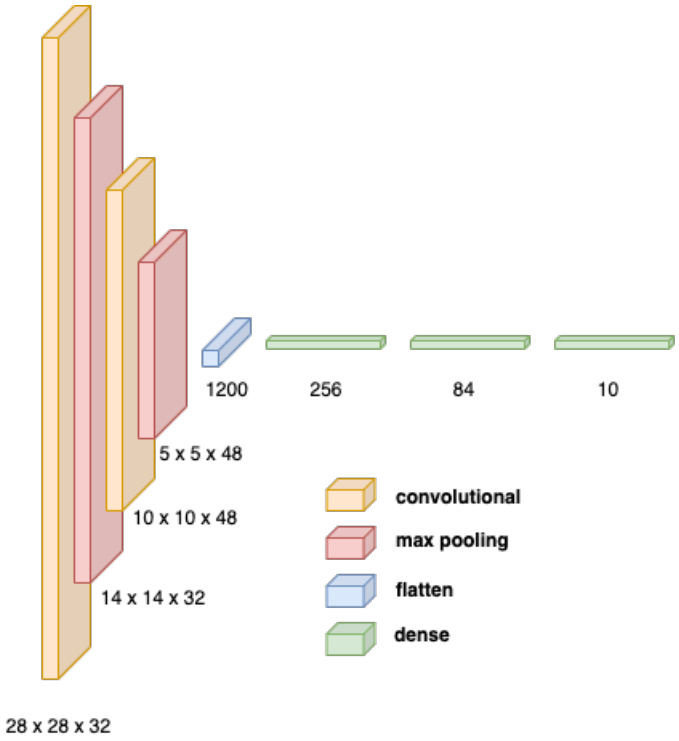


Figure 4.1.1: Model architecture of the developed CNN. The structure is based on the original LeNet-5 model presented by LeCun et al. (1998), and consists of two convolutional layers, two max-pooling layers, one flatten layer and three dense layers. Dropout is performed before each of the dense layers, with a dropout rate ($p = 0.3$) that was evaluated analytically by use of a validation set. The fact that this pre-existing dropout structure can be used to quantify uncertainty is the core feature of Monte Carlo Dropout. Otherwise, the relevant dimensions and parameters are presented underneath each layer of the network.

Initially, the trained model was used to perform inference on a sample from the MNIST dataset, belonging to the class 0. The generated prediction from the standard network is shown in terms of the softmax output from the final layer of the model, presented in the first row in Table 4.1.1. The model correctly classifies the sample as the digit 0, but does not, however, represent any uncertainty related to this prediction, as expected.

Subsequently, the same model was used to classify the sample, this time by using Monte Carlo Dropout with 100 forward passes through the network. The resulting prediction probabilities from this experiment are shown in the second row of Table 4.1.1. As shown by the classification probabilities, the model is still confident that the sample belongs to class 0, but is now able to provide additional information by means of a probability estimate in regards to the confidence of the inference. This uncertainty measure is obtained by averaging the predicted class probability for each of the classes obtained from the forward passes through the network. Due to the relative low complexity of the data and model, the upper limit of the recommended range of [30-100] stochastic forward passes provided by Gal and Ghahramani (2016) was used.

MNIST Label	0	1	2	3	4	5	6	7	8	9
Softmax Output	1	0	0	0	0	0	0	0	0	0
MCD Probability	0.996	0.0	0.002	0.0	0.0	0.0	0.001	0.001	0	0.001

Table 4.1.1: The softmax output produced by the initial CNN, as compared to the prediction probabilities produced by the MCD model. Both models correctly classify the sample as the digit 0, although only the MC Dropout model is able to provide a measure of uncertainty related to the inference.

Finally, a few practical notes in terms of the implementation are useful. In most cases, merely specifying the model's training mode to be true during inference is a sufficient implementation of Monte Carlo Dropout. Issues could arise, however, in scenarios where different parts of the model behave differently during inference and training, for example during batch normalization. Therefore, it is advisable to adopt the implementation as presented by Géron (2019) based on creating a custom layer that inherits from the regular Dropout feature, thus ensuring that dropout is switched on without interfering with any other parts, as intended. An example implementation of this in the Keras framework is presented in the below code snippet. This specific illustration is based on the Keras framework, but the overall logic and structure is applicable across different machine learning frameworks.

```
class MonteCarloDropout(keras.layers.Dropout):
    def call(self, inputs):
        return super().call(inputs, training = True)
```

Listing 4.1: Monte Carlo Dropout layer implemented in Keras..

Note that as dropout was already utilized as a regularization technique in the initial CNN, no adjustments to the model structure other than ensuring dropout to be true during inference were necessary in order to implement Monte Carlo Dropout.

4.2 Monte Carlo Dropout Regression Experiment

To explore the functionality of Monte Carlo Dropout within regression tasks, an experiment based on the *Boston Housing* dataset was performed. The Boston Housing dataset is derived from a study performed by the U.S. Census Service regarding the housing market in the Boston area, and provides 14 attributes for 506 distinct houses that can be used to predict the median value of a sample house, in $\$1000's$. As mentioned, uncertainty in regression tasks can be represented by a predictive mean and corresponding probability distribution.

A neural network consisting of three dense layers with ReLu non-linearities, with dropout applied with a rate of $p = 0.1$ between said layers, was developed for the experiment. The model architecture is inspired by the structure proposed for regression tasks by Gal and Ghahramani (2016), and is visualized in Figure 4.2.1.

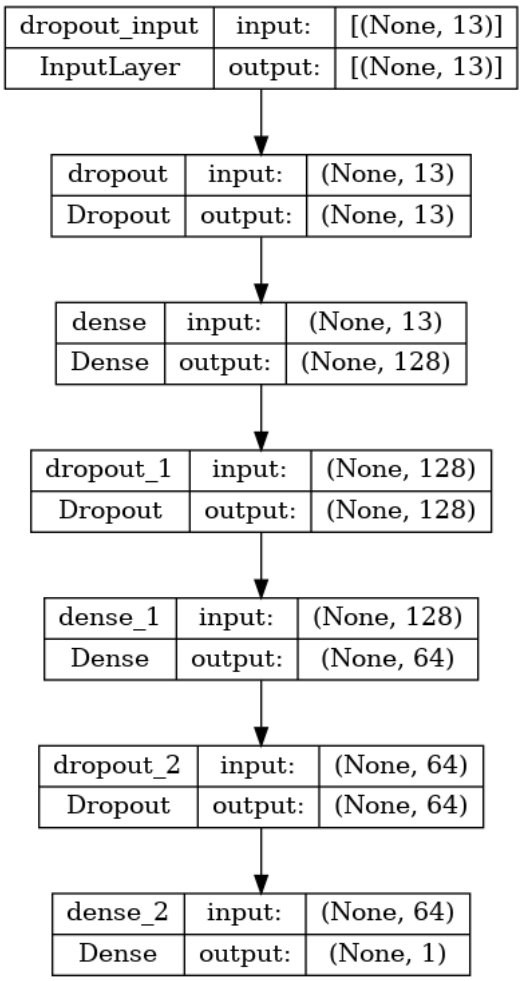


Figure 4.2.1: Model structure developed for the Monte Carlo Dropout regression task. The network is relatively simplistic, and consists of three dense layers with dropout applied between each of the layers. The dimensions of each layer is shown in the plot, and the dropout rate used is $p = 0.1$. The overall structure is inspired by Gal and Ghahramani (2016), and included to illustrate how a DNN employing dropout can be utilized to quantify uncertainty in a regression task.

After fitting the model to the training data, 100 passes through the network were performed on a sample consisting of 14 attributes, in order to generate a prediction regarding the median value of said sample. The results from the experiment are presented in Figure 4.2.2, in terms of a predicted mean and a corresponding density function around this mean. Once again, this amount of forward passes is feasible due to the low complexity of the data and model.

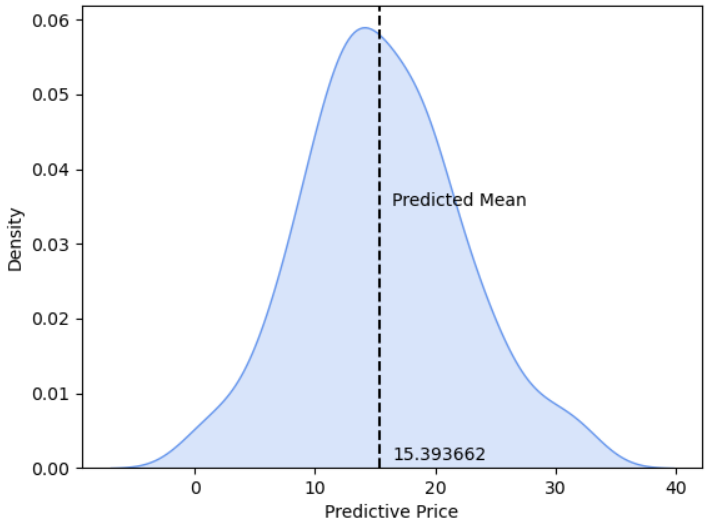


Figure 4.2.2: Result from performing 100 forward passes through the Monte Carlo Dropout model. The predicted mean, which represents the estimated median value of the sample in $\$1000's$, is provided with a corresponding probability density around said mean, as a measure of confidence in the regression estimate.

Practically, this is done by obtaining a predictive distribution through stochastic forward passes of the model, and then using the predicted mean as a point estimate. The results from this experiment show how adapting Monte Carlo Dropout in a regression model can generate a predictive mean and corresponding probability density, which provides a measure of uncertainty in terms of the generated predictions.

4.3 Softmax Probabilities on OOD Samples

To demonstrate the inadequacies of the softmax function in capturing OOD samples, a classification experiment based on augmented samples from the MNIST dataset was performed, the results of which are presented in Figure 4.3.1. The experiment was designed to demonstrate the softmax activation layer producing erroneously confident predictions when dealing with OOD samples, using the previously developed CNN without Monte Carlo Dropout, as presented in Figure 4.1.1.

Figure 4.3.1 details the classification probabilities produced by the model, utilizing the softmax activation layer, for a continuously rotated sample of a hand-drawn digit *1* from the MNIST benchmark dataset. As seen by the graphs, dependent on the angle of rotation of the digit, the model predicts the sample confidently as either *1*, *2* or *7*. As discussed in Section 4.3, the reason for this misrepresentation is the softmax function's inherent mathematical properties, as it is merely capable of presenting the probabilities in terms of their comparative value against the other modeled classes, which produces erroneously confident

predictions when OOD samples are introduced.

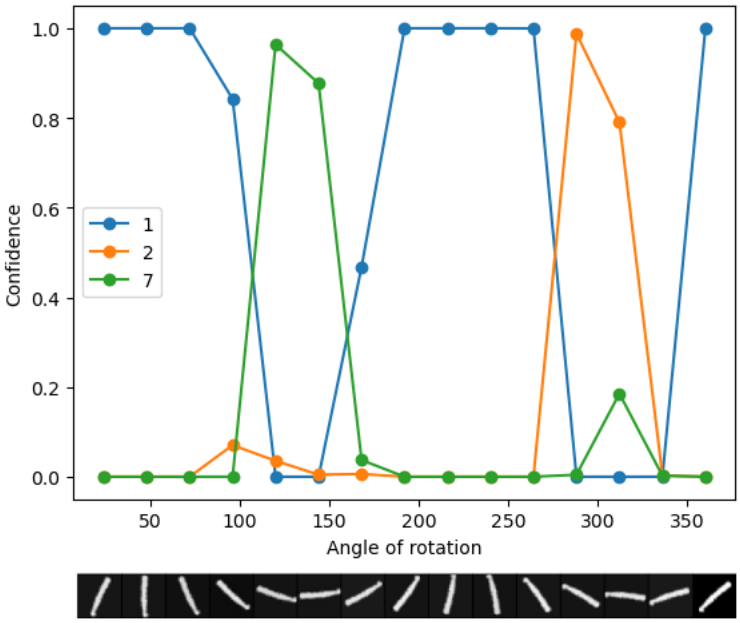


Figure 4.3.1: Demonstration of the softmax activation function's inability to capture OOD samples. Augmentations, in the form of rotations, performed on the digit *1* from the MNIST dataset result in the model producing erroneously confident predictions depending on the angle of said rotations. Especially looking at the ranges of rotation [100-175] and [275-350] degrees, it becomes evident that the confidence the model has in its predictions is misleading and how, therefore, softmax values should not be misconstrued as general model confidence. Note that the *confidence* measure in the graph refers to the softmax output of the model, and not overall model confidence, as discussed.

Although the softmax activation layer is an effective feature within class probability estimation for in-distribution samples, it is severely limited in terms of capturing out-of-distribution samples, as it treats each class as mutually exclusive. Approaches aimed as explicitly modeling a separate OOD-class have been proposed to amend for this limitation, although approaches such as Deep Ensembles have been shown to be a more effective method for capturing distributional shifts (Lakshminarayanan, Pritzel and Blundell 2016).

4.4 Deep Ensemble on OOD Samples

Deep Ensembles are widely considered an efficient and robust approach for capturing OOD samples that mitigates the deficiencies of merely using the softmax activation layer. To evaluate the effectiveness of this approach, a classification experiment based on the original paper by Lakshminarayanan, Pritzel and Blundell (2016) and the experiment presented in Section 4.3 was performed, the results from which are presented in Figure 4.4.1.

Firstly, the LeNet-5 inspired model, as presented in Figure 4.1.1, was fitted to the MNIST dataset and subsequently introduced to OOD samples from the notMNIST dataset, which are presented underneath the x-axis of the graph. The results from the single network's inferences are presented as the blue line, and represent the softmax probabilities related to the predicted class. Note that the predicted class labels are not presented, due to the fact that none of the samples belong to the data distribution to which the model has been fitted, and thus the model should ideally not be confident in any of it's predictions. As expected, however, the model is erroneously confident in it's predictions, even though it does not actually recognize the samples.

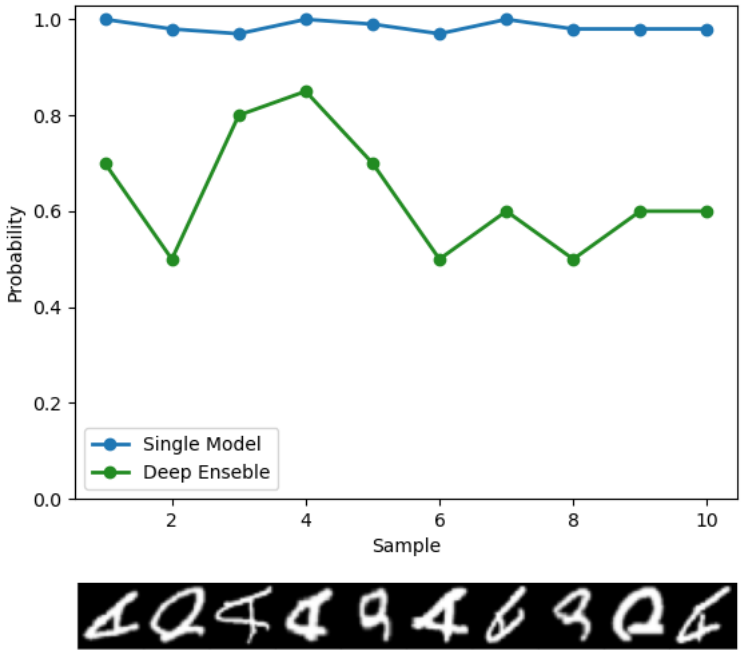


Figure 4.4.1: Experiment comparing a single network against an ensemble of 5 pseudo-randomly initialized instances of said network in terms of capturing OOD samples sampled from the notMNIST dataset, which are shown underneath the x-axis of the graph. The single network produces erroneously confident predictions and is not able to represent the fact that it has not been trained to recognize the provided samples. The ensemble, however, generates uncertainty-aware predictions that are more representative of the fact that the samples are not contained in the modeled data distribution. Note that the exact class the single network and ensemble predict are not presented, but rather the corresponding probability/confidence of the prediction itself. Due to the fact that none of the samples belong to the known data distribution, this information is sufficient in terms of demonstrating the erroneous nature of this representation.

The Deep Ensemble consists of 5 pseudo-randomly initialized instances of the single model fitted to the MNIST dataset, with the resulting classification probabilities being represented by the green line. The ensemble is based on the *randomization* approach presented by Lakshminarayanan, Pritzel and Blundell (2016),

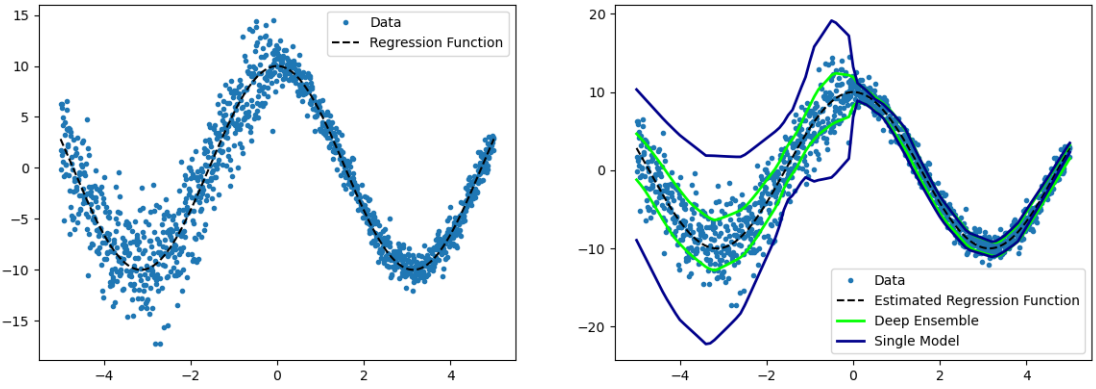
where the members of the ensemble are trained concurrently and independently. As mentioned, the optimal amount of members in an ensemble can be evaluated empirically, however, 5 members is widely recognized as a viable structure, which is also proposed as a sufficient amount by Lakshminarayanan, Pritzel and Blundell (2016). As is evident in the graph, the ensemble better represents the uncertainty that arises from being introduced to OOD samples, by providing an adjusted measurement of confidence in it's predictions, essentially quantifying the predictive uncertainty of the model.

The results show how generating an ensemble of neural networks provides an interpretable notion and representation of OOD samples, by averaging the predictions made by the underlying models. Flexibility exists in terms of the architecture of the ensemble, as well as the training and optimization procedures of these models, which is a major advantage of Deep Ensembles.

4.5 Deep Ensemble Regression Task

A regression experiment based on the initial paper by Lakshminarayanan, Pritzel and Blundell (2016) was performed to compare the performance of a Deep Ensemble to a single model on relatively simplistic toy data. Quasi-randomized data generated around a cosine function, as visualized in Figure 4.5.1a, was used in order to estimate the heteroscedastic aleatoric uncertainty around the estimated regression function. The estimated function is presented as the dark dotted line, whereas the upper and lower ranges of uncertainty for the single model and ensemble, measured by the variance around the estimated mean, are represented by the blue and green lines, respectively. The uncertainty in the range [-4, 0] was designed to be inherently higher than in the range [0, 4], by increasing the amount of outliers in the data as well as the general spread. A relatively simplistic dense neural network with ReLu nonlinearity was used for the experiment, the ensemble consisting of 5 instances of said network.

The experiment was performed to investigate the effect of combining several deep neural networks, as is the core principle of Deep Ensembles, compared to merely utilizing the single network, for a simplistic regression task. It is evident from Figure 4.5.1b that the ensemble produces more accurate uncertainty estimates, especially in terms of the high-uncertainty regions of the data. The main characteristic and benefit of the Deep Ensemble as compared to the single model, is it's robustness in terms of data noise and outliers as the ensemble is able to better generalize than the single model, which somewhat overfits to the particularities of the generated data.



(a) Data and Estimated Regression Function **(b)** Uncertainty Estimates

Figure 4.5.1: Deep Ensemble compared to a single model in capturing the uncertainty around an estimated regression function related to generated toy data. Quasi-randomized data was generated around a cosine function, with higher uncertainty in the range [-4, 0] than in the range [0, 4] in terms of outliers and general spread, as illustrated in a). The upper and lower bounds of the estimated uncertainty measured in terms of the variance around the estimated mean of the 5-network ensemble are presented as the green lines, whereas the single network is presented in blue.

This experiment shows how combining networks in an ensemble can provide better adjusted uncertainty estimates than merely using a single network in a regression task. The reasoning for this relates to Condorcet's jury theorem as presented in Section 3.2, as the ensemble is more robust and less prone to individual errors than a single network. As long as the additional infrastructure load that arises from hosting several models is feasible, Deep Ensembles are a viable and efficient approach for capturing uncertainty in regression tasks.

4.6 Temperature Scaling

To investigate the efficacy of Temperature Scaling an implementation of the 110-layer ResNet presented by He et al. (2016) was developed and trained on the CIFAR-100 dataset (Krizhevsky and G. Hinton 2009). Then, Temperature Scaling was used to calibrate the outputs of the network.

One general recommendation is to integrate Temperature Scaling directly into the training procedure, where subsequent to training the model the validation set can be used to obtain the temperature T. Then, this temperature can be used to re-scale the logit values within the softmax function, as presented in Figure 3.5.1. The re-calibrated softmax probability scores can then be used to convey the confidence levels of the predictions, which can be evaluated quantitatively through the use of reliability diagrams, as seen in Figure 4.6.1. These diagrams serve as a tool for visualizing the degree of alignment between two distributions, namely, the distributions of expected accuracy and predicted probabilities. Ideally, for a

perfectly calibrated network, these distributions should be as aligned as to generate a diagonal function across the diagram. Any deviation from this diagonal represents miscalibration in the network.

A standardized evaluation metric useful for assessing the efficacy of calibrations is the *expected calibration error* (ECE). This measure is presented by Guo et al. (2017), derived from the formulations presented by Pakdaman Naeini, Cooper and Hauskrecht (2015), and is based on dividing predictions into M distinct, equally-spaced bins. The formulation is derived from the discrepancy between the expected confidence and the accuracy, which, based on the average ratio of $\frac{accuracy}{confidence}$ from said bins, is defined in Equation (4.1), where n refers to the total number of samples. The partitioning of the predictions into distinct, equally-spaced bins mimics that of the histograms used in the reliability diagrams in Figure 4.6.1.

$$ECE = \Sigma_{m=1}^{M} \frac{|B_m|}{n} \; |acc(B_m) - conf(B_m)| \tag{4.1}$$

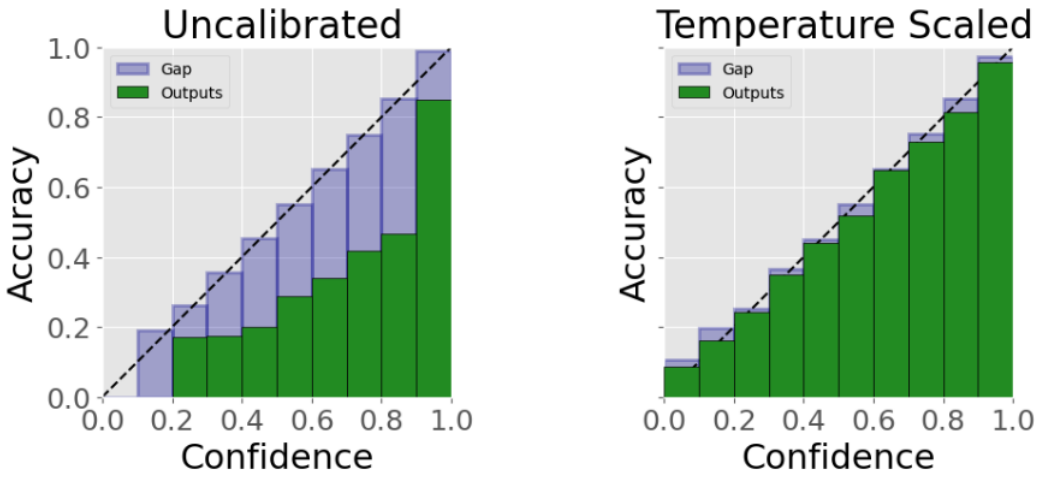
The measures of accuracy and confidence are presented in Equations (4.2) and (4.3), respectively, where B_m refers to the set of indices of samples whose prediction confidence falls within the interval $I_m = (\frac{m-1}{M}, \frac{m}{M})$, based on M interval bins sized $\frac{1}{M}$.

$$acc(B_m) = \frac{1}{|B_m|} \Sigma_{i \in B_m} \mathbf{1}(\hat{y}_i = y_i) \tag{4.2}$$

Here, $\mathbf{1}$ represents an indicator function that takes the value 1 should the related statement be true, and 0 otherwise. In this case, this refers to scenarios in which the predicted and actual labels are equal, meaning the prediction is correct. Also, y and $\hat{y}$ refers to the true and predicted class label, respectively, whereas $\hat{p}_i$ represents the confidence of a sample i.

$$conf(B_m) = \frac{1}{|B_m|} \Sigma_{i \in B_m} \hat{p}_i \tag{4.3}$$

Using the aforementioned measures as evaluation metrics, the effects of employing the post-hoc calibration method can be visualized in the reliability diagram in Figure 4.6.1.



(a) ECE = 18.7 **(b)** ECE = 2.1

Figure 4.6.1: Reliability diagrams before and after calibration of a 110-layer ResNet fitted to the CIFAR-100 dataset. The expected calibration error (ECE) before and after calibration is presented underneath each diagram. As seen in a) the uncalibrated model tends to be overconfident in its predictions, whereas the Temperature Scaled model in b) closely resembles the idealized diagonal function, which is represented by the dotted black line.

In practical terms the temperature can be set to $T = 1$ during training before being optimized in terms of the negative log-likelihood on the validation set. After this, the obtained logits can be multiplied by a parameter $\frac{1}{T}$ in the softmax layer. Note that using the same validation set for the training and the calibration is a prerequisite of Temperature Scaling.

The results demonstrate how simply re-scaling the logit scores using the validation set can combat overconfident predictions and generate better confidence estimates. Even though each bin is merely adjusted by a single scalar value, the overall calibration is well-adjusted.

CHAPTER

FIVE

DISCUSSION

A useful supplement to the presented theory is a discussion and reflection surrounding the potential advantages and disadvantages of the different methods. The following section will present this, alongside a general overview of which considerations should be made when implementing a framework for quantifying uncertainty in a deep learning model.

5.1 Discussion

5.1.1 General Considerations

When considering which uncertainty quantification method to employ for an application there is a wide range of factors to consider. Generally, the specific requirements and necessary considerations are highly application-specific, meaning a definitive guide and answer as to which exact model and method is optimal for a specific scenario does not necessarily exist. Even so, an informed decision can be made based on certain underlying guidelines and factors, ranging from *computational efficiency*, *storage requirements*, *scalability*, *accuracy*, *interpretability*, *robustness*, and *complexity of implementation*. Understanding the specific requirements and constraints of the task at hand is a major part of the machine learning field, of which the user has to make an informed decision, based on certain underlying factors, often multi-variable and complex.

One thing to note is that the presented methods have primarily been experimented with on benchmark datasets and tasks in the available literature, in order to provide a notion of the method's generalized performance. However, as with most machine learning methods, tailoring the model for application-specific use should prove beneficial in terms of performance. That is to say that any presented results should not deter one from experimenting with implementing any of the methods for application-specific use, as the explicit tailoring of the models and methods for any specific application could greatly improve the performance. Additionally, fine-tuning of hyperparameters will considerably impact the model's overall performance. This means that independent of the choice of model and uncertainty quantification method, parameter tuning is necessary to obtain high-quality results and should be considered as a factor independent of the model and

method selection.

In many real-world applications a necessary feature of any viable machine learning model is the ability to dynamically adapt to changes in the data distribution over time. Historically, many deep learning systems have been based on the presupposition that the test and training data share the same distribution. Due to the dynamic nature of the world and the physical phenomena that these systems are concerned with however, the validity of this presupposition is somewhat constrained, as future data tends to steadily diverge from the established distribution over time. Consequently, the reliability of future predictions can falter. One critical issue that has been exposed and discussed by Ovadia et al. (2019), Fort, Hu and Lakshminarayanan (2019) and Nalisnick et al. (2019) is that even though the predictive performance of a model deteriorates over time following this shift in data distribution, the corresponding measures of confidence increase, essentially leading to a concealed failure. On the basis of this phenomenon, some mechanism for capturing data shifts and detecting OOD samples are deemed critical for any viable deep learning system.

As per Guo et al. (2017), despite its simplicity, Temperature Scaling has been shown to improve both the accuracy and calibration of classification models without the need for intrusive model adjustments. The findings from the experiment performed in Section 4.6 support this notion, by illustrating how the method can be used to calibrate a trained classification model, supposing it is based on a softmax activation layer. Employing the method can help identify whether a model is overly confident or uncertain in its predictions, which combined with its low computational requirements, relative simplicity of implementation and effectiveness, means it should be considered a useful supplement to any classification model. Another advantage of the method is that the running time is highly efficient, as it scales linearly with the amount of validation set samples, and can essentially be seen as a one-dimensional convex optimization problem (Guo et al. 2017). Although several network calibration methods exist, the team claims that Temperature Scaling is *"the simplest, fastest, and most straightforward of the methods, and surprisingly is often the most effective"*.

Generally, it is natural to distinguish Monte Carlo Dropout and Deep Ensembles from Prior Networks and Evidential Deep Learning considering certain factors and features. Structurally it is natural to make this distinction, in addition to the differences related to interpretability, computational complexity and specialized knowledge requirements. Both Monte Carlo Dropout and Deep Ensembles are well established and recognized methods within the field, whereas Prior Networks and Evidential Deep Learning are relatively new approaches, which differ mainly in the fact that they attempt to explicitly incorporate uncertainty into the model structure itself. Prior Networks and Evidential Deep Learning are generally considerably more complex than Monte Carlo Dropout and Deep Ensembles in regard to implementation, interpretability and optimization, and require domain-specific knowledge and competence. In highly application-specific cases however, should thorough, pre-existing knowledge regarding the problem domain be available, these methods might be desirable. In terms of the required competence and

pre-requisite experience of the user, Prior Networks and Evidential Deep Learning are also more demanding, which is another contributing factor to recommending Monte Carlo Dropout and Deep Ensembles as alternative solutions.

The aforementioned reasoning is the rationale behind selecting Monte Carlo Dropout and Deep Ensembles as the focus for the practical implementations in Section 4. Prior Networks and Evidential Deep Learning are included in the thesis to showcase an alternative avenue of research beyond traditional approaches. These methods show great promise, and future contributions could help further advance their development and applications.

5.1.2 Monte Carlo Dropout and Deep Ensembles

Deep Ensembles and Monte Carlo Dropout differ in one fundamental aspect, namely in terms of the underlying nature of the distributions they generate over potential functions that fit the data. Generally, the parameters of a neural network outnumber the amount of training samples that the model is fitted to by a significant amount, meaning, in theory, there exists many different potential functions that approximate the data-generating function. As such, there exists many low-loss valleys/minima and regions in the theoretical loss landscape, which correspond to different viable candidate functions. By combining and ensembling several of these functions from a Deep Ensemble, through the pseudo-random initialization of the underlying models, the higher the robustness and likeliness of this ensemble of functions to accurately portray the underlying data distribution is, which enables the model to represent the desired low confidence associated with data points not contained in the initial distribution, otherwise known as out-of-distribution samples. By doing so, the ensemble obtains a distribution comprised of diverse functions, located at different low-loss regions (Fort, Hu and Lakshminarayanan 2019).

Monte Carlo Dropout, however, differs in a slight but crucial way. Most Bayesian approaches inherently tend toward one single low-loss valley or region in the loss landscape, from which it draws its distribution of candidate functions. Due to this fact, the generated distribution from Monte Carlo Dropout and other Bayesian approaches are comprised of somewhat similar functions, in regards to the theoretical data-space as a whole, which means the model is not as robust or fitted for capturing out-of-distribution samples. Essentially, ensembles can be said to be better approximators of the true predictive distributions, as there is more variety in the underlying candidate functions, which can make them more robust and reliable.

The main advantage of Monte Carlo Dropout is its comparatively low complexity and ease of implementation compared to the other methods, as well as its applicability within the field of transfer learning. As discussed, no adjustments to the architecture/structure or retraining of the initial model are necessary, as the predictions are merely drawn from the predictive distribution obtained through stochastic forward passes. Also, as these forward passes can be run concurrently, the computational efficiency of the method is relatively high. Being able to ex-

ploit previously trained and optimized models is highly advantageous, reducing the workload dramatically. The main challenge when implementing the method is evaluating the optimal dropout rate, based on the complexity of the model and amount of data, among other factors. Different techniques exist for this, but generally, empirically evaluating the value by using a validation set is the recommended practice. Monte Carlo Dropout might not be feasible for highly complex models, as the computational cost of computing multiple forward passes during inference is more expensive than traditional deterministic predictions.

Deep Ensembles is an alternative approach to traditional Bayesian methods that is relatively simplistic in terms of implementation, requires little hyperparameter tuning, is parallelizable, and provides high quality uncertainty estimates. Another benefit of the approach is that it is to a large extent model-independent, meaning flexibility exists in designing and developing the ensembles' underlying neural networks. Deep Ensembles perform well in capturing out-of-distribution samples, and are able to capture both aleatoric and epistemic uncertainty. The main drawback of the approach is that it can be resource-demanding, as it requires the training, storing and optimizing of several models. The ensembles consist of M times the amount of parameters compared to a single network, which means that in certain memory-constrained applications, the storage requirements might be limiting. In such cases, however, the ensemble can be distilled into a simpler model, as per Bucila, Caruana and Niculescu-Mizil (2006) and G. Hinton, Vinyals and Dean (2015b).

Also, it is worth to mention that for small tasks with little available data or highly specialized tasks, Deep Ensembles have not be shown to be as effective. This is somewhat due to the fact that, as previously discussed, the strength of ensembles stems from generating uncertainty estimates obtained from drawing candidate functions across a large loss landscape. For a small task with few data points this landscape is inherently less diverse, in which the additional complexity of hosting, training and optimizing several distinct models might not be warranted in terms of the overall performance. In such a scenario, Monte Carlo Dropout should be preferable.

5.1.3 Prior Networks and Evidential Deep Learning

One of the major challenges of utilizing Prior Networks is that they require explicitly defining and tuning a suitable prior distribution, which can be difficult in practice. Also, the complexity of development, implementation and optimization is higher than for Monte Carlo Dropout and Deep Ensembles, alongside the requirement of specialized knowledge on the part of the user. The main advantage of the method is that domain-specific knowledge can be implemented directly into the model structure, which means that less data and less extensive training procedures might be necessary, as the model does not need to learn and infer the previously modeled knowledge. In addition, one crucial distinction and additional feature of Prior Networks is the fact that they aim to explicitly capture *distributional uncertainty*, by parameterizing a prior distribution of the predictive

distributions. Traditionally, distributional uncertainty has been encapsulated in the overall model or data uncertainty, where as Malinin and Gales (2018) claim that this further distinction and categorization is useful, as a means of explicitly capturing and identifying a mismatch between the domains of the observed data and the training data.

Evidential Deep Learning, as proposed by Sensoy, Kaplan and Kandemir (2018), is able to capture both aleatoric and epistemic uncertainty in classification tasks, and attained attain state-of-the-art results within two specific tasks, namely detection of OOD queries, as well as endurance against adversarial perturbations. In a similar fashion to Prior Networks, Evidential Deep Learning might potentially shorten and reduce the required training procedure, by explicitly incorporating knowledge into the model structure. The main challenges in terms of Evidential Deep Learning relate to the complexity of development, implementation and optimization. Evidential Deep Learning can be less computationally efficient than Prior Networks, as the method involves estimating full probability distributions, as opposed to estimating point estimates of uncertainty. The interpretability is challenging, as understanding the underlying rationale of decisions and designing the model structure and underlying probability distributions requires some potentially esoteric knowledge on the part of the user. Even though the probabilistic representation provided by Evidential Deep Learning in the form of probability distributions offer a more comprehensive representation of the uncertainty, interpreting this information might require expert knowledge and additional interpretation techniques. Prior Networks, on the other hand, provide less demanding uncertainty estimates in the form of a variance and standard deviation.

The potentially extensive training and development procedures of the methods can be limiting as opposed to standard deep learning approaches, especially in terms of the fine-tuning of hyperparameters. Also, these models are highly sensitive to the choice of the prior distribution, which adds to the overall complexity of the model. The methods are more intrusive than Monte Carlo Dropout and Deep Ensembles, as more often than not they require restructuring and altering of the underlying model structure, which further adds to the potential complexity of the approaches. The methods show promise as novel approaches for quantifying uncertainty and have been included with the intention of including an avenue for future research and progress that offers an alternative perspective to traditional methods, although, due to the aforementioned factors, they are not as feasible for real-world applications as of yet.

5.1.4 Summary

Although no definite statement can be provided in regard to the optimal model and method selection independent of the task at hand, certain general guidelines and conclusions can be drawn. Generally speaking, Monte Carlo Dropout and Deep Ensembles are natural staring points for any endeavor within uncertainty quantification. Both methods have been proven effective for quantifying uncertainty, which, along with their relative simplicity, interpretability, and computational ef-

ficiency make them desirable as opposed to Prior Networks and Evidential Deep Learning, which, although promising, are more complex and demand more specialized knowledge and skills.

Monte Carlo Dropout is desirable over Deep Ensembles in two distinct scenarios, namely within transfer learning and should computational complexity and memory resources be limited. Utilizing pre-existing trained models is often highly advantageous, which is possible for Monte Carlo Dropout, as the method does not require restructuring the underlying model architecture. Also, the computational complexity and memory requirements of Deep Ensembles are higher, as they require the storing and optimizing of multiple models. Therefore, if hosting multiple models poses a concern due to the additional infrastructure load, Monte Carlo Dropout is a reasonable alternative. Otherwise, Deep Ensembles have been shown to be well equipped for quantifying uncertainty and capturing OOD samples, as proven by the state-of-the-art results obtained by Ovadia et al. (2019).

In classification models Deep Ensembles can be combined with Temperature Scaling, by calibrating each of the underlying models of the ensemble. By doing so, the individual models that comprise the ensemble are well adjusted, which reduces the overall overconfidence of the ensemble and produces more reliable uncertainty estimates. Temperature Scaling is valuable when interpreting uncertainty estimates in deep learning models, as the method offers a way to calibrate the model's confidence by adjusting the temperature, thereby enhancing the interpretability and reliability of the uncertainty estimates.

Should explicitly distinguishing and capturing the uncertainty that arises from data shifts be desirable, Prior Networks are a potential approach. As opposed to the other methods, Prior Networks are designed to explicitly capture so-called *distributional uncertainty*, which stems from the uncertainty related to a mismatch between the distributions of the observed data and the training data. Therefore, the approach is favorable if this feature is particularly desirable for any application.

5.2 Future Work and Literature Gaps

The main field of future interest within uncertainty quantification methods is *scalability*. Many of the available methods are severely limited by their computational complexity, especially when dealing with large amounts of data and complex models. In order for these methods to be feasible in real-world applications, the issue of scalability needs to be accounted for. In concordance with this, the computational efficiency of existing and future methods is an area of vital importance.

Another area of future research is further and more deeply investigating the underlying sources of uncertainty. Most contemporary methods are primarily concerned with distinguishing between aleatoric (data), epistemic (model) uncertainty, and potentially the distributional uncertainty. Even further categorization

could be possible, however, for example parameter uncertainty and structural uncertainty. Accurately identifying the underlying cause of the uncertainty facilitates for efficient managing through implementation of specific countermeasures.

An important aspect of machine learning models is *interpretability*. Essentially, the interpretability of a machine learning model refers to the ability to understand and comprehend how and why the model's decisions are made, in a way that is understandable to humans. This is especially important in safety-critical applications, for the transparency, accountability and trustworthiness of the models, in addition to facilitating for further improvements and correcting of biases in the data and model, as well as detecting inconsistencies and errors. Also, providing insight into the relationships between input features and predictions can help developers further improve the models. Following the surge in development of machine learning methods and models it is crucial to recognize the importance of interpretability, and concurrently develop means and methods of interpreting the models.

The focal point of attention of uncertainty quantification in deep learning has previously been directed toward supervised learning tasks. Future work should include areas such as *reinforcement learning*, as accurate uncertainty estimates are important for effective decision-making and exploration-exploitation trade-offs. Application and implementation of the methods within areas of machine learning such as long-short-term-memory (LSTM) networks, transformers and self-supervised/semi-supervised learning should be further explored in the future, following the relevancy and surge in the prevalence of large language models (LLM).

As a means of further improving the effectiveness and robustness of Monte Carlo Dropout, *advanced dropout variants* should be investigated in future work. This refers to adaptive or task-specific dropout strategies for dynamically adjusting the dropout rates, based on the complexity of the data or the specific requirements and characteristics of the task. Algorithms and methods for counteracting the computational burden associated with stochastic forward passes through a network should be investigated, some of which include leveraging hardware accelerators, approximate inference methods, or developing better sampling strategies that can obtain accurate uncertainty estimates with fewer forward passes. Also, the adaption of Monte Carlo Dropout to structures such as Recurrent Neural Networks (RNN) and LSTM networks is a potential future application of the method. These structures correlate to the ideas underlying Gaussian Process Dynamical Models and Recursive Gaussian Processes, which means that one would expect there to exist a suitable dropout approximation for these structures as well. Finally, exploring the effects of hybrid approaches, where Monte Carlo Dropout is combined with other uncertainty quantification methods such as Deep Ensembles should be more deeply explored in future work.

An active area of research within Deep Ensembles is called *implicit ensembles*, which refers to structures where the members of an ensemble share model parameters, by use of methods such as *snapshot ensembles* (Huang et al. 2017), *multiple heads* (S. Lee, Purushwalkam et al. 2015; Osband et al. 2016) and *swapout*

(Singh, Hoiem and Forsyth 2016). Briefly stated, these methods are based on sharing weights during training, before implicitly ensembling them during inference, which could prove more efficient than traditional ensembles and might mitigate some of the storage limitations previously encountered. Most of the current research within Deep Ensembles focuses on random initialization of the weights of the independent networks of the ensemble, in order to obtain distinct loss trajectories across the models. Little research exists in terms of Deep Ensembles within transfer learning, however, where the goal is to reuse pre-existing trained model weights.

Another promising approach within Deep Ensembles is based on explicitly decorrelating the predictions made by the underlying networks, with the intention of promoting ensemble diversity to improve the overall performance of the ensemble. This approach is presented by S. Lee, Purushwalkam Shiva Prakash et al. (2016), and should be further explored within the context of uncertainty quantification in the future. Two additional features that show promise in terms of improving the performance of Deep Ensembles are *stacking* (Wolpert 1992) and *adaptive mixture of experts* (Jacobs et al. 1991), which are both methods designed for optimizing ensemble weights.

As Prior Networks and Evidential Deep Learning are novel approaches within the field there is a general lack of literature on the effectiveness of the methods within a range of applications, such as natural language processing (NLP), speech recognition, reinforcement learning and computer vision. Currently, the main limitation of these methods stems from challenges related to scalability and computational requirements, especially for complex data and models. Better algorithms and methods for improving the computational efficiency and scalability should be developed, some of which include *approximate inference*, *adaptive sampling*, and *model compression techniques*. Another possible avenue of future work is to combine Prior Networks with traditional Bayesian approaches, in order to exploit the strengths of each of the approaches. Approaches such as *variational inference* and *Markov Chain Monte Carlo Dropout* are potential developments within this concept. Also, the effect of Prior Networks and Evidential Deep Learning in different real-world applications should be investigated in future research, in order to assess the viability of the methods and identify potential specific challenges and requirements that arise in different contexts.

CHAPTER

SIX

CONCLUSION

The thesis has provided a conceptual overview and introduction to uncertainty quantification in deep learning, by presenting the underlying theoretical foundations, alongside contemporary methodologies from the existing literature. In addition to this, experiments have been performed in order to investigate how the theoretical concepts relate to practical implementation. The presented work aims to serve as a useful resource for any endeavor of academic or industrial nature, by offering readers a general overview of the field.

For classification models a general recommendation is to use *Deep Ensembles* combined with *Temperature Scaling* when quantifying uncertainty. Two notable exceptions exist however, namely within transfer learning scenarios and should the infrastructure load of hosting multiple models be a concern. In these scenarios *Monte Carlo Dropout* is recommended as an alternative. Deep Ensembles have been shown to improve accuracy, uncertainty estimates and out-of-distribution robustness in deep learning models, which, combined with its flexibility, interpretability and relative simplicity of implementation makes it a viable approach for many uncertainty quantification tasks. Note that Temperature Scaling is only applicable for classification models utilizing a softmax activation layer, and that for regression tasks the same recommendation applies, except the post-hoc calibration.

Generally, based on the findings of the pioneering authors and the available literature, $m = 5$ members in an ensemble and a range of [30-100] stochastic forward passes through the neural network for Monte Carlo Dropout is recommended as a general guideline. The optimal value of both of these parameters, however, is application-specific, and usually depends on the desired trade-off between computational cost and accuracy, and can be evaluated analytically by using a validation set. In practice, the single scalar parameter T used to calibrate networks in Temperature Scaling can be set to $T = 1$ during training, and subsequently be optimized in terms of the negative log-likelihood on the validation set. Note that a prerequisite of Temperature Scaling is that the same validation set is used for the training and the calibration.

REFERENCES

Abdar, Moloud et al. (2021). 'A review of uncertainty quantification in deep learning: Techniques, applications and challenges'. In: *Information Fusion* 76, pp. 243–297.

Aggarwal, Charu C. et al. (2014). 'Active learning: A survey'. In: *Data Classification*. CRC Press. DOI: 10.1201/b17320.

Amini, Alexander et al. (2019). 'Deep Evidential Regression'. In: DOI: 10.48550/ARXIV.1910.02600. URL: https://arxiv.org/abs/1910.02600.

Ashukha, Arsenii et al. (2020). 'Pitfalls of In-Domain Uncertainty Estimation and Ensembling in Deep Learning'. In: *arXiv preprint arXiv:2002.06470*.

Bao, Wentao, Qi Yu and Yu Kong (2021). 'Evidential Deep Learning for Open Set Action Recognition'. In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 13329–13338. DOI: 10.1109/ICCV48922.2021.01310.

Blahut, Richard E. (2002). '25 - Information Theory and Coding'. In: *Reference Data for Engineers (Ninth Edition)*. Ed. by Wendy M. Middleton and Mac E. Van Valkenburg. Ninth Edition. Woburn: Newnes, pp. 25-1-25–31. ISBN: 978-0-7506-7291-7. DOI: https://doi.org/10.1016/B978-075067291-7/50027-3. URL: https://www.sciencedirect.com/science/article/pii/B9780750672917500273.

Blei, David M., Andrew Y. Ng and Michael I. Jordan (03/2003). 'Latent Dirichlet Allocation'. In: *J. Mach. Learn. Res.* 3.null, pp. 993–1022. ISSN: 1532-4435.

Blundell, Charles et al. (2015). *Weight Uncertainty in Neural Networks*. DOI: 10.48550/ARXIV.1505.05424. URL: https://arxiv.org/abs/1505.05424.

Bucila, Cristian, Rich Caruana and Alexandru Niculescu-Mizil (08/2006). 'Model compression'. In: vol. 2006, pp. 535–541. DOI: 10.1145/1150402.1150464.

Choi, Sungjoon et al. (2017). *Uncertainty-Aware Learning from Demonstration using Mixture Density Networks with Sampling-Free Variance Modeling*. DOI: 10.48550/ARXIV.1709.02249. URL: https://arxiv.org/abs/1709.02249.

Council, National Research (2012). *Assessing the Reliability of Complex Models: Mathematical and Statistical Foundations of Verification, Validation, and Uncertainty Quantification*. Washington, DC: The National Academies Press. ISBN: 978-0-309-25634-6. DOI: 10.17226/13395. URL: https://nap.nationalacademies.org/catalog/13395/assessing-the-reliability-of-complex-models-mathematical-and-statistical-foundations.

Davis, Josiah, Jason Zhu and Jeremy Oldfather (2020). 'Quantifying uncertainty in deep learning systems.' In: *AWS*.

Deng, Li (2012). 'The mnist database of handwritten digit images for machine learning research'. In: *IEEE Signal Processing Magazine* 29.6, pp. 141–142.

Depeweg, Stefan et al. (2018). *Decomposition of Uncertainty in Bayesian Deep Learning for Efficient and Risk-sensitive Learning*. arXiv: 1710.07283 [stat.ML].

DeVries, Terrance and Graham W. Taylor (2018). *Leveraging Uncertainty Estimates for Predicting Segmentation Quality*. DOI: 10.48550/ARXIV.1807.00502. URL: https://arxiv.org/abs/1807.00502.

Dietterich, Thomas G. (2000). 'Ensemble Methods in Machine Learning'. In: *Multiple Classifier Systems, First International Workshop, MCS 2000, Cagliari, Italy, June 21-23, 2000, Proceedings*. Ed. by Josef Kittler and Fabio Roli. Vol. 1857. Lecture Notes in Computer Science. Springer, pp. 1–15. DOI: 10.1007/3-540-45014-9_1. URL: https://doi.org/10.1007/3-540-45014-9%5C_1.

Ebden, Mark (2015). *Gaussian Processes: A Quick Introduction*. DOI: 10.48550/ARXIV.1505.02965. URL: https://arxiv.org/abs/1505.02965.

Estlund, David M. (1994). 'Opinion leaders, independence and Condorcet's Jury Theorem'. In: *Theory and Decision* 36, pp. 131–162. DOI: https://doi.org/10.1007/BF01079210.

Filos, Angelos et al. (2019). *A Systematic Comparison of Bayesian Deep Learning Robustness in Diabetic Retinopathy Tasks*. DOI: 10.48550/ARXIV.1912.10481. URL: https://arxiv.org/abs/1912.10481.

Fort, Stanislav, Huiyi Hu and Balaji Lakshminarayanan (2019). *Deep Ensembles: A Loss Landscape Perspective*. DOI: 10.48550/ARXIV.1912.02757.

Gal, Yarin (2016). 'Uncertainty in Deep Learning'. In: *Department of Engineering, University of Cambridge*.

— (2015). 'What My Deep Model Doesn't Know'. In: URL: https://www.cs.ox.ac.uk/people/yarin.gal/website/blog_3d801aa532c1ce.html.

Gal, Yarin and Zoubin Ghahramani (2016). *Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning*. arXiv: 1506.02142 [stat.ML].

Gelman, Andrew et al. (2014). *Bayesian Data Analysis*. URL: http://www.stat.columbia.edu/~gelman/book/BDA3.pdf.

Géron, Aurélien (2019). In: *Hands-On Machine Learning with Scikit-Learn, Keras and TensorFlow, 2nd Edition*. Vol. 31. O'Reilly Media, Inc.

Goodfellow, I. J. (2015). 'Explaining and harnessing adversarial examples'. In: *ICLR*.

Görtler, Jochen, Rebecca Kehlbeck and Oliver Deussen (2019). 'A Visual Exploration of Gaussian Processes'. In: *Distill*. https://distill.pub/2019/visual-exploration-gaussian-processes. DOI: 10.23915/distill.00017.

Guo, Chuan et al. (2017). *On Calibration of Modern Neural Networks*. DOI: 10.48550/ARXIV.1706.04599. URL: https://arxiv.org/abs/1706.04599.

Harper, Ross and Joshua Southern (04/2022). 'A Bayesian Deep Learning Framework for End-To-End Prediction of Emotion From Heartbeat'. In: *IEEE Transactions on Affective Computing* 13.2, pp. 985–991. DOI: 10.1109/taffc.2020.2981610. URL: https://doi.org/10.1109%2Ftaffc.2020.2981610.

He, Kaiming et al. (2016). 'Deep Residual Learning for Image Recognition'. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778. DOI: 10.1109/CVPR.2016.90.

Heendeni, J. N. et al. (2008). 'A Generalization of Bayesian Inference in the Dempster-Schafer Belief Theoretic Framework'. In: *Springer*.

Hendrycks, Dan and Kevin Gimpel (2018). *A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks*. arXiv: 1610.02136 [cs.NE].

Hinton, Geoffrey, Oriol Vinyals and Jeff Dean (2015a). *Distilling the Knowledge in a Neural Network*. DOI: 10.48550/ARXIV.1503.02531. URL: https://arxiv.org/abs/1503.02531.

— (2015b). *Distilling the Knowledge in a Neural Network*. arXiv: 1503.02531 [stat.ML].

Hochreiter, Sepp and Jürgen Schmidhuber (12/1997). 'Long Short-term Memory'. In: *Neural computation* 9, pp. 1735–80. DOI: 10.1162/neco.1997.9.8.1735.

Houlsby, Neil et al. (2012). 'Collaborative Gaussian processes for preference learning'. In: *NIPS*.

Huang, Gao et al. (2017). *Snapshot Ensembles: Train 1, get M for free*. arXiv: 1704.00109 [cs.LG].

Hüllermeier, Eyke and Willem Waegeman (03/2021). 'Aleatoric and epistemic uncertainty in machine learning: an introduction to concepts and methods'. In: *Machine Learning* 110.3, pp. 457–506. DOI: 10.1007/s10994-021-05946-3.

Jacobs, Robert et al. (02/1991). 'Adaptive Mixture of Local Expert'. In: *Neural Computation* 3, pp. 78–88. DOI: 10.1162/neco.1991.3.1.79.

Jaynes, E. T. (05/1957). 'Information Theory and Statistical Mechanics'. In: *Phys. Rev.* 106 (4), pp. 620–630. DOI: 10.1103/PhysRev.106.620. URL: https://link.aps.org/doi/10.1103/PhysRev.106.620.

Jøsang, Audun (2016). 'Subjective Logic: A Formalism for Reasoning Under Uncertainty'. In: *Springer*.

Jungo, Alain, Richard McKinley et al. (09/2018). 'Towards Uncertainty-Assisted Brain Tumor Segmentation and Survival Prediction'. In: pp. 474–485. ISBN: 978-3-319-75237-2. DOI: 10.1007/978-3-319-75238-9_40.

Jungo, Alain, Raphael Meier et al. (2018). *Uncertainty-driven Sanity Check: Application to Postoperative Brain Tumor Cavity Segmentation*. DOI: 10.48550/ARXIV.1806.03106. URL: https://arxiv.org/abs/1806.03106.

Kandemir, Melih (2015). 'Asymmetric Transfer Learning with Deep Gaussian Processes'. In: *Proceedings of the 32nd International Conference on Machine Learning*. Vol. 37. Proceedings of Machine Learning Research. PMLR, pp. 730–738.

Kapoor, Ashish et al. (2010). 'Gaussian Processes for Object Categorization'. In: *International Journal of Computer Vision* 88.2, pp. 169–188. DOI: 10.1007/s11263-009-0268-3.

Kendall, Alex and Yarin Gal (2017). *What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?* DOI: 10.48550/ARXIV.1703.04977.

Kim, H. and J. Lee (2007). 'Clustering Based on Gaussian Processes'. In: *MIT Press - Journals* 19.11, pp. 3088–3107. DOI: 10.1162/neco.2007.19.11.3088.

Kingma, Diederik P., Tim Salimans and Max Welling (2015). *Variational Dropout and the Local Reparameterization Trick*. DOI: 10.48550/ARXIV.1506.02557. URL: https://arxiv.org/abs/1506.02557.

Kotz, Samuel, Balakrishnan Narayanaswamy and Norman L. Johnson (2000). 'Models and Applications, 2nd Edition'. In: Volume 1.

Krizhevsky, Alex and Geoffrey Hinton (2009). 'Learning multiple layers of features from tiny images'. In: *Computer Science Department, University of Toronto, Tech. Rep* 1.4.

Krizhevsky, Alex, Ilya Sutskever and Geoffrey E. Hinton (2012). 'Imagenet classification with deep convolutional neural networks'. In: *NIPS*.

Lakshminarayanan, Balaji, Alexander Pritzel and Charles Blundell (2016). 'Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles'. In: DOI: 10.48550/ARXIV.1612.01474.

LeCun, Y et al. (1998). 'Gradient-based learning applied to document recognition'. In: *Proceedings of the IEEE* 86.11, pp. 2278–2324.

LeCun, Yann, Yoshua Bengiuo and Geoffrey Hinton (2015). *Deep Learning*. DOI: https://doi.org/10.1038/nature14539.

Lee, Stefan, Senthil Purushwalkam et al. (2015). *Why M Heads are Better than One: Training a Diverse Ensemble of Deep Networks*. arXiv: 1511.06314 [cs.CV].

Lee, Stefan, Senthil Purushwalkam Shiva Prakash et al. (2016). 'Stochastic Multiple Choice Learning for Training Diverse Deep Ensembles'. In: *Advances in Neural Information Processing Systems*. Ed. by D. Lee et al. Vol. 29. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper_files/paper/2016/file/20d135f0f28185b84a4cf7aa51f29500-Paper.pdf.

Leibig, Christian et al. (2017). 'Leveraging uncertainty information from deep neural networks for disease detection'. In: *bioRxiv*. DOI: 10.1101/084210.

Malinin, Andrey, Sergey Chervontsev et al. (2020). *Regression Prior Networks*. URL: https://openreview.net/forum?id=ygWoT6hOc28.

Malinin, Andrey and Mark Gales (2018). *Predictive Uncertainty Estimation via Prior Networks*. DOI: 10.48550/ARXIV.1802.10501. URL: https://arxiv.org/abs/1802.10501.

Mikolov, Tomas et al. (2013). *Efficient Estimation of Word Representations in Vector Space*. arXiv: 1301.3781 [cs.CL].

Mitchell, Tom M. (1997). *Machine Learning*. McGraw-Hill International Editions Computer Science Series.

Mitchell, Tom Michael (2002). 'The Need for Biases in Learning Generalizations'. In: URL: https://www.researchgate.net/publication/2534848_The_Need_for_Biases_in_Learning_Generalizations.

Nalisnick, Eric et al. (2019). *Do Deep Generative Models Know What They Don't Know?* arXiv: 1810.09136 [stat.ML].

Nguyen, Vu-Linh, Sébastien Destercke and Eyke Hüllermeier (2019). *Epistemic Uncertainty Sampling*. DOI: 10.48550/ARXIV.1909.00218. URL: https://arxiv.org/abs/1909.00218.

Nix, D. A. and A. S. Weigend (1994). 'Estimating the mean and variance of the target probability distribution'. In: *IEEE International Conference on Neural Networks*.

— (2015). 'Explaining and harnessing adversarial examples'. In: *ICLR*.

Norouzi, Alireza et al. (05/2019). 'Exploiting Uncertainty of Deep Neural Networks for Improving Segmentation Accuracy in MRI Images'. In: pp. 2322–2326. DOI: 10.1109/ICASSP.2019.8682530.

Osband, Ian et al. (2016). 'Deep Exploration via Bootstrapped DQN'. In: *Advances in Neural Information Processing Systems*. Ed. by D. Lee et al. Vol. 29. Curran

Associates, Inc. URL: https://proceedings.neurips.cc/paper_files/paper/2016/file/8d8818c8e140c64c743113f563cf750f-Paper.pdf.

Ovadia, Yaniv et al. (2019). *Can You Trust Your Model's Uncertainty? Evaluating Predictive Uncertainty Under Dataset Shift*. DOI: 10.48550/ARXIV.1906.02530. URL: https://arxiv.org/abs/1906.02530.

Pakdaman Naeini, Mahdi, Gregory Cooper and Milos Hauskrecht (2015). 'Obtaining Well Calibrated Probabilities Using Bayesian Binning'. In: vol. 29. DOI: 10.1609/aaai.v29i1.9602. URL: https://ojs.aaai.org/index.php/AAAI/article/view/9602.

Pichler, Georg et al. (2022). *A Differential Entropy Estimator for Training Neural Networks*. arXiv: 2202.06618 [cs.LG].

Platt, John C. (1999). 'Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods'. In: *Advances in large margin classifiers*. MIT Press, pp. 61–74.

Quiñonero-Candela, Joaquin et al. (2010). 'Dataset Shift in Machine Learning'. In: *Journal of the Royal Statistical Society. Series A (Statistics in Society)* 173. DOI: 10.2307/20622600.

Rasmussen, Carl Edward and Christopher K. I. Williams (11/2005). *Gaussian Processes for Machine Learning*. The MIT Press. ISBN: 9780262256834. DOI: 10.7551/mitpress/3206.001.0001. URL: https://doi.org/10.7551/mitpress/3206.001.0001.

Roy, Abhijit Guha et al. (2018). *Bayesian QuickNAT: Model Uncertainty in Deep Whole-Brain Segmentation for Structure-wise Quality Control*. arXiv: 1811.09800 [cs.CV].

Sensoy, Murat, Lance Kaplan and Melih Kandemir (2018). 'Evidential Deep Learning to Quantify Classification Uncertainty'. In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio et al. Vol. 31. Curran Associates, Inc.

Sentz, Karl and Scott Ferson (04/2002). 'Combination of Evidence in Dempster-Shafer Theory'. In: DOI: 10.2172/800792. URL: https://www.osti.gov/biblio/800792.

Shi, Yuge (2019). 'Gaussian Processes, not quite for dummies'. In: *The Gradient*.

Singh, Saurabh, Derek Hoiem and David Forsyth (2016). *Swapout: Learning an ensemble of deep architectures*. arXiv: 1605.06465 [cs.CV].

Soch, Joram and Carsten Allefeld (2016). 'Kullback-Leibler Divergence for the Normal-Gamma Distribution'. In: *arXiv: Statistics Theory*.

Srivastava, Nitish et al. (2014). 'Dropout: A Simple Way to Prevent Neural Networks from Overfitting.' In: *Journal of Machine Learning Resarch* 15, pp. 1929–1958.

Szegedy, C. (2014). 'Intriguing properties of neural networks'. In: *JMLR*.

Tousignant, Adrian et al. (2019). 'Prediction of Disease Progression in Multiple Sclerosis Patients using Deep Learning Analysis of MRI Data'. In: *International Conference on Medical Imaging with Deep Learning*.

Tran, Dustin, Rajesh Ranganath and David M. Blei (2015). *The Variational Gaussian Process*. DOI: 10.48550/ARXIV.1511.06499. URL: https://arxiv.org/abs/1511.06499.

Turner, Richard E. (11/2016). 'Gaussian Processes: From the Basics to the State-of-the-Art'. In: Computational and Biological Learning Lab, Department of

Engineering, University of Cambrindge. URL: http://cbl.eng.cam.ac.uk/pub/Public/Turner/News/imperial-gp-tutorial.pdf.

Vandal, Thomas et al. (2018). 'Prediction and uncertainty quantification of daily airport flight delays'. In: International Conference on Predictive Applications and APIs.

Varshney and Hima Alemzadeh (2017). 'On the safety of machine learning: Cyber-physical systems, decision sciences, and data products.' In: *CoRR*.

Wang, Guotai et al. (2019). 'Aleatoric uncertainty estimation with test-time augmentation for medical image segmentation with convolutional neural networks'. In: *Neurocomputing* 338, pp. 34–45. DOI: https://doi.org/10.1016/j.neucom.2019.01.103.

Welling, Max and Yee Whye Teh (2011). 'Bayesian Learning via Stochastic Gradient Langevin Dynamics'. In: Proc. International Conference on Machine Learning (ICML).

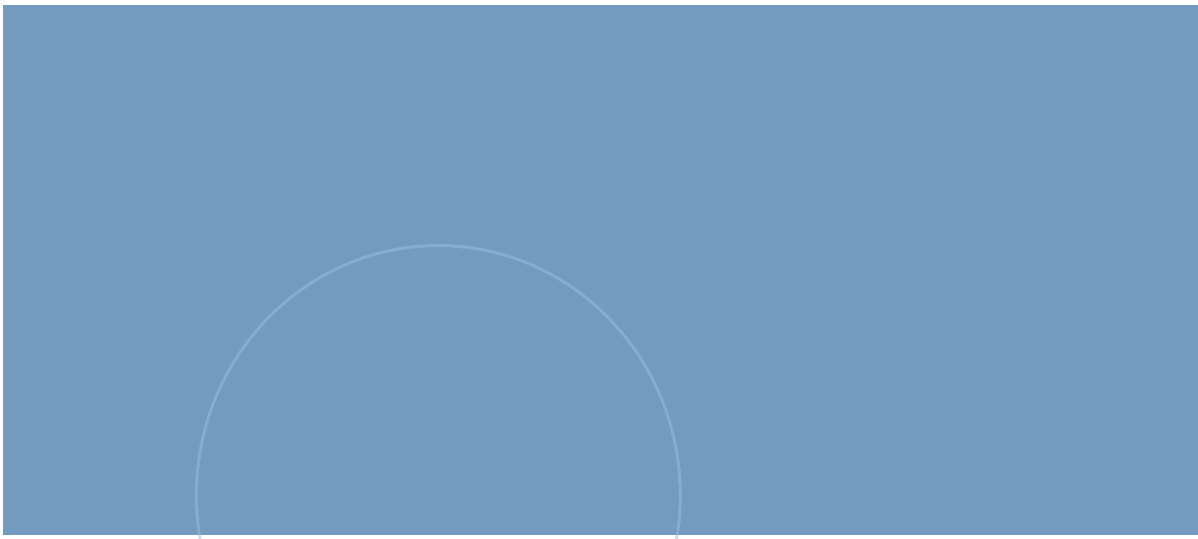
Wickstrøm, Kristoffer, Michael Kampffmeyer and Robert Jenssen (2018). 'Uncertainty modeling and interpretability in convolutional neural networks for polyp segmentation'. In: IEEE International Workshop on Machine Learning for Signal Processing.

Wilson, Andrew Gordon et al. (2016). 'Deep Kernel Learning'. In: *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*. Vol. 51. Proceedings of Machine Learning Research. PMLR, pp. 370–378.

Witten, Ian H., Eibe Frank and Mark A. Hall (2016). *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Series in Data Management Systems.

Wolpert, David (12/1992). 'Stacked Generalization'. In: *Neural Networks* 5, pp. 241–259. DOI: 10.1016/S0893-6080(05)80023-1.

Zhou, Jian and Olga G. Troyanskaya (2015). 'Predicting effects of noncoding variants with deep learning-based sequence model'. In: *Nature Methods*.



NTNU
Norwegian University of
Science and Technology