ASSESSING THE RELIABILITY OF DEEP NEURAL NETWORKS

Dissertation
zur Erlangung des Grades eines

DOKTORS DER INGENIEURWISSENSCHAFTEN

der Technischen Universität Dortmund
an der Fakultät für Informatik

von

PHILIPP OBERDIEK

Dortmund

2023

Tag der mündlichen Prüfung:	06.09.2023
Dekan:	Prof. Dr.-Ing. Gernot A. Fink
Gutachter:	Prof. Dr.-Ing. Gernot A. Fink
	Prof. Dr. Stefan Harmeling

ACKNOWLEDGMENTS

At the beginning of my bachelor's degree, I could not have imagined that one day I would write a dissertation. But here I am now writing the acknowledgments for my doctoral thesis. However, this would not have been possible without the support of many people.

First and foremost, I need to thank my supervisor Prof. Dr.-Ing. Gernot A. Fink. Thank you, Gernot, for the continuous inspiration, countless discussions and mentoring you offered throughout the last years. I also need to thank you for the opportunity to pursue this path and the great freedom I had in developing my interests and research direction. I would also like to thank Prof. Dr. Stefan Harmeling for writing the second review of my thesis and for the participation in the doctoral committee. Furthermore, for taking the role as a mentor and shaping many of the publications in this work, I would like to thank Dr. Matthias Rottmann. Matthias, thank you very much for numerous discussions and your continuous commitment to getting the best out of me. In the role of being my official mentor in the structured doctoral program, I would like to thank Prof. Dr. Gabriele Kern-Isberner. Thank you for sharing your experience and a different perspective on my work. Also, I would like to thank Prof. Dr. Peter Buchholz and Prof. Dr. Emmanuel Müller for their participation in the doctoral committee.

I am also especially grateful for the people that I was able to work with in the pattern recognition group. There, I would like to especially mention Fabian Wolf, Fernando Moya, Arthur Matei, Kai Brandenbusch, Dominik Koßmann, Oliver Tüselmann, Nilah Ravi Nair, Eugen Rusakov, Tim Raven and Dr.-Ing. Leonhard Rothacker. Without you, the time in the office would have been not as fun. Countless after-lunch sessions chatting about research, teaching and other stuff. Grabbing a cup of coffee and sharing a good joke and laugh together. The conference and summer-school trips are also something special to remember as is our monthly team event. All in all definitely something to miss there, and I can proudly say that I found new friends during this time. I would also like to thank Prof. Dr.-Ing. Gernot A. Fink and Dr. Robin Chan for their valuable feedback to my thesis manuscript.

Last but not least I would like to thank my family and especially my wife Jennifer. You celebrated all my successes with me and encouraged me during more difficult phases. At times, you suffered with me and that's why I can say that you deserve a good part of this success.

CONTENTS

NOTATION

$a, b, c, \ldots$	scalar
$\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, \ldots$	vector
$\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \ldots$	matrix or tensor
$\mathcal{A}, \mathcal{B}, \mathcal{C}, \ldots$	set
$a^{(1)}, a^{(2)}, a^{(3)}, \ldots$	multiple elements from the same set
$\lvert\mathcal{A}\rvert$	cardinality of set $\mathcal{A}$
$\boldsymbol{W}_i$	i-th row of matrix $\boldsymbol{W}$
$\boldsymbol{W}_{\cdot j}$	j-th column of matrix $\boldsymbol{W}$
$\boldsymbol{W}_{ij}$	element of matrix $\boldsymbol{W}$ at i-th row and j-th column
$\boldsymbol{x}_i$	i-th element of vector $\boldsymbol{x}$
$d_{\boldsymbol{x}}$	dimensionality of $\boldsymbol{x}$
$\mathbb{I}_d$	unit matrix of size $d \times d$
X, Y	random variables
$P(X, Y)$	probability distribution over X and Y
$p(x, y)$	density for $x \sim X$ and $y \sim Y$
$p(y \mid x)$	conditional probability
$\hat{p}(y \mid x)$	estimated conditional probability
$\mathbb{E}_p[X]$	expected value of X w.r.t. density p
$\boldsymbol{\theta}$	vector of parameters
$f(x)$, $f(\boldsymbol{x})$	scalar function accepting scalar or vector inputs
$\boldsymbol{f}(\boldsymbol{x})$	vector function accepting vector inputs
$\boldsymbol{f}(\boldsymbol{x})_i$	i-th element of the vector function output
$f'(x)$	derivative of f w.r.t. input argument x
$f(x \mid \boldsymbol{w})$	function which requires $\boldsymbol{w}$ to be given
$\log(x)$	logarithm of x, if not specified otherwise uses base e
$\lVert\cdot\rVert_{\mathrm{p}}$	L_{p}-norm
$\mathbb{1}_{\{\mathrm{expr}\}}$	indicator function being 1 if expr is true and 0 otherwise
$\nabla_{\boldsymbol{\theta}}\mathfrak{L}$	gradient of $\mathfrak{L}$ w.r.t. $\boldsymbol{\theta}$
$f * g$	convolution of two functions f and g

If a function f is only defined with a support of $\mathbb{R}$, then applying this function to a vector $\boldsymbol{x}$ implicates an element-wise application of f. Vectors with special meaning might receive clarifying information in the exponent, e.g., $\boldsymbol{y}^{\mathrm{oh}}$ or $\boldsymbol{\theta}^{\mathrm{MAP}}$. Additionally, there might be multiples of a matrix, tensor or vector, designating similar things, which will also be indicated by an index in the exponent, e.g., $\boldsymbol{W}^1$ or ϕ^l. Scalars may also be enumerated with a lower index, e.g., n_1, n_2 and n_3.

1 INTRODUCTION

Deep Neural Networks (DNNs) have achieved astonishing results in the last two decades, fueled by ever larger datasets and the availability of high performance compute hardware. This led to breakthroughs in many applications such as image and speech recognition, natural language processing, autonomous driving, and drug discovery. Despite their success, the understanding of internal workings and the interpretability of predictions remains limited and DNNs are often treated as "black boxes".

Especially for safety-critical applications where the well-being of humans is at risk, decisions based on predictions should consider associated uncertainties. Autonomous vehicles, for example, operate in a highly complex environment with potentially unpredictable situations that can lead to safety risks for pedestrians and other road users. In medical applications, decision based on incorrect predictions can have serious consequences for a patient's health.

As a consequence, the topic of Uncertainty Quantification (UQ) has received increasing attention in recent years. The goal of UQ is to assign uncertainties to predictions in order to ensure the decision-making process is informed by potentially unreliable predictions. In addition, other tasks such as identifying model weaknesses, collecting additional data or detecting malicious attacks can be supported by uncertainty estimates.

Unfortunately, UQ for DNNs is a particularly challenging task due to their high complexity and nonlinearity. Uncertainties which can be derived from traditional statistical models are often not directly applicable to DNNs. Therefore, the development of new UQ techniques for DNNs is of paramount importance to ensure safety-aware decision-making.

This thesis evaluates existing UQ methods and proposes improvements and novel approaches which contribute to the reliability and trustworthiness of modern deep learning methodology. A comprehensive and consistent evaluation of several UQ tasks and a diverse selection of datasets provide insights into the strengths and weaknesses of existing methods and the proposed approaches. The contributions of this thesis and where they were previously published are summarized in Section 1.1, after which an outline of the remainder of the thesis is given.

1.1 CONTRIBUTIONS

Gradient Metrics for Detecting Out-of-Distribution Inputs

DNNs tend to have far more learnable parameters than samples in a common training set, and are therefore susceptible to large amounts of model uncertainty (more on that in Chapters 2 and 3). For practical applications it is thus of utmost importance to be able to reliably model this source of uncertainty. While Bayesian Neural Networks (BNNs) offer a theoretically grounded way to compute model uncertainty, they also require a change in the learning procedure and add a significant amount of computational effort during inference, which often makes them impractical. Gradient metrics belong to the external frequentist approaches and therefore can be applied to pre-trained models without requiring any change in the training procedure or architecture of the Neural Network (NN). They were first proposed by Oberdiek, Rottmann, and Gottschalk [100] in the context of Out-of-Distribution (OoD) and False Positive (FP) detection. Since then, gradient metrics were adopted in the research community and applied to a variety of UQ problems. In this thesis the gradient metrics are evaluated against a more diverse benchmark and compared to recent advances in the field of UQ. The author's contributions to the original publication [100] are the choice of auxiliary loss, gradient aggregation metrics, meta classification models for summarizing multiple gradient statistics, experimental evaluation and writing of the manuscript with supporting reviews from the co-authors.

Framework for Retrieval Based Exploration of Unknown Objects

Detecting unknown objects during inference is an important step towards applying DNNs in safety critical applications. However, as the physical world is under constant change, it is inevitable to also update our model continuously. UQ offers a way to identify unknown object classes and augment the training dataset with new examples. A periodic re-training on the updated dataset ensures that the NN can keep up with this constant environmental changes. Oberdiek, Rottmann, and Fink [99] proposed a retrieval based exploration of unknown objects in the context of semantic segmentation of street scenes. By gathering segments with a high predictive uncertainty and embedding them in a low-dimensional semantic space, a user is able to search this space based on a Query by Example (QbE). In this work ([99]), the author contributed the general idea of an Out-of-Distribution retrieval framework, exploration support based on clustering techniques, experimental evaluation, and writing of the manuscript with supporting reviews from the co-authors. Additionally, the author of this thesis published an open source software tool for applying the framework to arbitrary semantic segmentation applications.[1]

Generative Model for Complete Uncertainty Quantification

Besides the model uncertainty, aleatoric or data uncertainty is equally important to quantify during the deployment phase of an NN. Reliable estimates of aleatoric uncertainty can be used to detect ambiguous inputs and misclassifications, and the distinction between different sources of uncertainty can be valuable information for active learning applications and interpretability. Oberdiek, Fink, and Rottmann [98] proposed a new One-versus-All (OvA) classification model which is integrated into the training of a conditional Generative Adversarial Network (GAN). The loss functions are combined in such a way that the generative model produces examples that cover the boundary of the training distribution, therefore shielding each class from the rest of the data space. Previous works exist that formulate similar loss functions for generating boundary samples [79, 135] but generating class-conditional Out-of-Class (OoC) examples is a novel contribution of [98]. This approach fits exceptionally well with an OvA classifier and is able to distinguish between aleatoric and epistemic uncertainty. In addition to outstanding results on the task of OoD detection, outperforming many previous works in this field, the generated data also improves the overall model accuracy. A mode collapse problem of the generator, which was observed in previous works by other authors, was solved by applying a low-dimensional regularizer based on the cosine-similarity. Although related publications used regularization techniques within the input data space, the application to a low-dimensional data representation in combination with the cosine-similarity is novel and another contribution of this work. The contributions of the author of this thesis to [98] are the evaluation of ideas in the direction of generative model based OoD detection, building of the conditional generative learning pipeline within a low-dimensional representation space, transferring previous works on boundary generation with Jensen-Shannon (JS) GANs to a Wasserstein based loss formulation, the low-dimensional regularization loss based on the cosine-similarity, parts of the theory to the OvA classifier, an extensive qualitative and quantitative evaluation, and writing of the manuscript with supporting reviews from the co-authors. Additionally, the author published an open access and extensible code base to evaluate the proposed frameworks as well as 8 related methods on a variety of uncertainty tasks.[2]

1 https://github.com/RonMcKay/OODRetrieval
2 https://github.com/RonMcKay/UQGAN

Evaluating Uncertainty Quantification Methods for Adversarial Example Detection

While there are publications analyzing the adversarial example detection performance of uncertainty quantification approaches [44, 123] and methods specialized on adversarial example detection [3], the evaluation is mostly limited to a few approaches and only carried out on small datasets like MNIST and CIFAR10. This thesis will additionally contribute a larger evaluation on 4 datasets (MNIST, CIFAR10, CIFAR100 and Tiny ImageNet) and compare the detection performances between 12 uncertainty quantification methods (including 4 approaches described in this thesis). From each class of uncertainty quantification method (see Chapter 4), there will be at least one competitor evaluated on 3 common adversarial attacks. As all methods are reimplemented and applied to the same dataset splits, a fair and consistent performance evaluation is guaranteed. In summary, the evaluation in this thesis improves in two dimensions over previous works, the number of UQ methods compared and the diversity of data selection.

1.2 OUTLINE

This thesis is split into 9 chapters, with the current chapter containing the introduction, contributions and outline. The remaining 8 chapters are summarized in this section.

Chapter 2 - Uncertainty in Machine Learning
This chapter introduces the notion of uncertainty in the context of machine learning. The basic concepts of statistical learning theory are explained in Section 2.1. In the following Section 2.2 different sources and types of uncertainty are described. Additionally, we understand how to measure uncertainties based on probability distributions and learn about model assumptions which separate uncertainty sources. The chapter concludes by explaining a workflow for controlling shifts in learning environments that uses uncertainty estimates.

Chapter 3 - Neural Networks
The theory of DNNs is introduced in this chapter. After a brief review of the beginnings of NNs, the standard training algorithm is derived using a Multilayer Perceptron (MLP). We gain insights into connections between Maximum Likelihood Estimation (MLE) and the training of NNs, which aligns with the distribution based uncertainty quantification described in Section 2.2. Following, the concept of Convolutional Neural Networks (CNNs) and basic building blocks of modern DNNs are introduced. In the last section, a short overview over the history of DNN architectures and a more detailed description of the models used in the experiments is given. While explaining the algorithms and architectural choices of NNs, there will be comments and explanations whenever these are causing specific types of uncertainties.

Chapter 4 - Related Work

An overview of the literature landscape of UQ, adversarial examples and application areas of uncertainties is given in this chapter. UQ methods are categorized into 4 main categories and the most relevant publications of each are reviewed. This is followed by a summary of algorithms which can generate adversarial examples. Afterwards, there will be an overview of tasks which can utilize uncertainty estimates. The chapter is concluded by summarizing advantages and disadvantages of the reviewed UQ approaches.

Chapter 5 - Gradient Metrics

A computationally inexpensive and external approach for the quantification of epistemic uncertainty, called *Gradient Metrics* is proposed in this chapter. The idea behind the method as well as the theory is explained. This is followed by a review of similar works which emerged in recent years.

Chapter 6 - Uncertainty Quantification GAN

Here, an approach for quantifying different types of uncertainty based on GANs is proposed. In the beginning we will motivate different design choices and compare them to existing generative models. Subsequently, the theory of an OvA classification model is derived, and we learn how the model is able to separate different types of uncertainties. The aforementioned OvA classifier is then embedded into a generative learning framework for which the theory is explained in Section 6.2. A low-dimensional regularizer is motivated and explained in the remainder of the chapter.

Chapter 7 - Evaluating Uncertainty

In Chapter 7 different downstream tasks for evaluating the quality of uncertainty estimates are described. Afterwards, common evaluation metrics for binary classification models are defined and explained.

Chapter 8 - Experiments

The second-to-last chapter describes qualitative and quantitative results for a variety of UQ methods and datasets. All related methods and the proposed approaches are benchmarked on 4 tasks and each subsection concludes with the findings of the quantitative evaluation.

Chapter 9 - Conclusion

The thesis is concluded with the main highlights and findings and discusses potential future work.

After the main part of the thesis, additional quantitative results on the uncertainty downstream tasks and a study of hyperparameters applicable to the proposed approaches are given. The bibliography and a list of acronyms complete the document. All references with associated URLs were last checked for content and availability on September 13, 2023.

2 UNCERTAINTY IN MACHINE LEARNING

This chapter is going to introduce the basic notion of uncertainty in the context of machine learning. After the basic concepts of statistical learning theory are explained in Section 2.1, different sources and types of uncertainties are introduced in Section 2.2. Afterwards, we will understand how uncertainties can be computed based on probability distribution and learn about model assumptions which separate uncertainty sources.

2.1 STATISTICAL LEARNING THEORY

The ideas and notation in this section are based on the work by Shalev-Shwartz and Ben-David [119].

The goal of statistical learning theory is to mathematically describe the process of automatic learning from existing data in order to gain knowledge and reason about newly arising data. This is often called Machine Learning (ML) and modern algorithms such as DNNs can also be classified under this term.

Let us start by defining our learning-setting. We define the domain from which we are trying to learn as $\mathcal{X}$. In the case of, e.g., gray scale images of size 32×32, we can say that $\mathcal{X} = \{0, 1, \dots, 255\}^{32\times 32}$. Although an element of $\mathcal{X}$ can have many dimensions, we will refer to an element of $\mathcal{X}$ as a vector $\boldsymbol{x}$ to simplify notation. This thesis is predominantly focusing on the problem of *supervised* learning, in which the learner is supplied with samples $(\boldsymbol{x}, y) \in \mathcal{D} = \mathcal{X} \times \mathcal{Y}$ during training. Data from $\mathcal{X}$ is collected and annotated, which results in the training dataset $\mathcal{S} = \{(\boldsymbol{x}^{(1)}, y^{(1)}), \dots, (\boldsymbol{x}^{(N)}, y^{(N)})\} \subseteq \mathcal{D}$. The goal is then to learn a mapping $h : \mathcal{X} \to \mathcal{Y}$, resembling the true underlying labeling function. We will, unless stated otherwise, describe $\mathcal{S}$ as a set in contrast to Shalev-Shwartz and Ben-David [119], who describe it as a sequence. The data points in $\mathcal{S}$ are usually assumed to be independent and identically distributed (i.i.d.) according to some joint probability distribution $P(X, Y)$ on $\mathcal{D}$ with its corresponding probability mass function $p(\boldsymbol{x}, y)$. We also call $P(X, Y)$ the data generating distribution. As we are focusing on classification problems in the rest of this thesis, let us define $\mathcal{Y} = \{1, \dots, K\}$, meaning each data point $(\boldsymbol{x}, y) \in \mathcal{S}$ can be assigned to one of K classes. In the case of classifying gray scale images of handwritten digits we would set $\mathcal{Y} = \{0, 1, \dots, 9\}$. Lastly, we define the set of all possible mappings $h : \mathcal{X} \to \mathcal{Y}$ that our learning algorithm can produce as our *hypothesis space* $\mathcal{H}$. Analogously, a realization h of our learning algorithm is called a *hypothesis*.

We measure the success of a hypothesis h with a loss function $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ by computing the expected loss (also called risk) over $\mathcal{D}$ (see also [61])

$$R(h) := \int_{\mathcal{D}} \ell(h(\boldsymbol{x}), y) \,\mathrm{d}P(\boldsymbol{x}, y)\,. \tag{2.1}$$

One example for ℓ is the $0/1$-Loss defined as

$$\ell^{0/1}(h(\boldsymbol{x}), y) = \mathbb{1}_{\{h(\boldsymbol{x}) \neq y\}}\,. \tag{2.2}$$

Equation (2.1) is infeasible to compute as $\mathcal{D}$ usually has an infinite number of elements and the data generating distribution $P(X, Y)$ is not known. This is why in practice we compute the *empirical risk* over our training dataset $\mathcal{S}$

$$R_{\text{emp}}(h) := \frac{1}{N} \sum_{(\boldsymbol{x},y) \in \mathcal{S}} \ell(h(\boldsymbol{x}), y)\,. \tag{2.3}$$

We then find our hypothesis by

$$\hat{h} = \underset{h \in \mathcal{H}}{\arg\min}\, R_{\text{emp}}(h)\,. \tag{2.4}$$

In this case $\hat{h}$ is called Empirical Risk Minimizer (ERM). If it holds that

$$R_{\text{emp}}(h) \xrightarrow[N \to \infty]{} R(h)\,, \tag{2.5}$$

we can say that our learner is consistent. In practice however, our training dataset $\mathcal{S}$ is of finite size and the ERM will always be only an approximation of the true risk minimizer

$$h^* := \underset{h \in \mathcal{H}}{\arg\min}\, R(h)\,. \tag{2.6}$$

In real-world applications (e.g., image classification), considering h to be a deterministic mapping is not a reasonable assumption. As $\mathcal{D}$ usually has an infinite number of elements, we are always depending on our training dataset $\mathcal{S}$ for the approximation of the true underlying sampling distribution. We are assuming that $\mathcal{S}$ is representative of $\mathcal{X}$ and all examples $(\boldsymbol{x}^{(i)}, y^{(i)})$ are i.i.d., but this assumption may be wrong due to, e.g., small sample size or domain shifts. If $\mathcal{S}$ is only covering a subset of $\mathcal{X}$, there remain examples that our learning algorithm has never seen before. Fixing on a single $\hat{h} \in \mathcal{H}$ could lead to a hypothesis that gives incorrect predictions for all (possibly infinitely many) examples of $\mathcal{X}$ which are not covered by $\mathcal{S}$. If we define the mapping $\mathcal{X} \to \mathcal{Y}$ as being non-deterministic, we are not solving the problem of missing data, but we enable the model to express uncertainty about a prediction. In this case our model covers a subset of $\mathcal{H}$, which results in the mapping $\boldsymbol{h} : \mathcal{X} \to [0,1]^{|\mathcal{Y}|}$, with $\sum_{i=1}^{K} \boldsymbol{h}(\boldsymbol{x})_i = 1$. Additionally, the loss function needs to be defined as $\ell : [0,1]^K \times [0,1]^K \to \mathbb{R}$ and our label y can be seen as a realization of a categorical distribution. An example for such a loss function over distributions is the Kullback-Leibler Divergence (KLD) (more on that in Section 3.3.1).

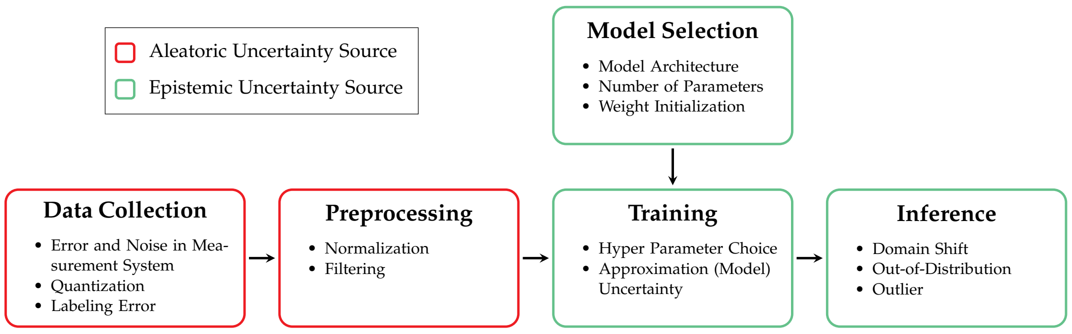

Figure 2.1: Flow diagram of a machine learning pipeline, summarizing different sources of uncertainty.

2.2 SOURCES OF UNCERTAINTY

Malinin and Gales [89], Hüllermeier and Waegeman [61] and Gawlikowski et al. [38, p. 3-4] describe different sources of uncertainty in the context of statistical learning theory and machine learning. Generally, uncertainty can be distinguished into *aleatoric* and *epistemic* uncertainty. The different types can be attributed to steps in a machine learning pipeline as illustrated in Fig. 2.1.

The aleatoric uncertainty refers to the irreducible part of the total uncertainty, resulting in a non-deterministic mapping between $\mathcal{X}$ and $\mathcal{Y}$ and producing a distribution over possible outcomes

$$p(y \mid \boldsymbol{x}) = \frac{p(\boldsymbol{x}, y)}{p(\boldsymbol{x})} . \tag{2.7}$$

This means that even in the case of complete knowledge about the true underlying sampling distribution $p(\boldsymbol{x}, y)$, there still remains uncertainty that can not be reduced. This can be a result of noise in the data collection process (e.g., noise produced by an image sensor), loss of information due to quantization of the data, labeling errors, class overlap, or simply completely random effects inherent to the environment. Consider, e.g., the problem of weather forecasting. In the best case we would have information about every atomic particle in the universe and it's interactions between one another. Even in this hypothetical example, Heisenberg's uncertainty principle teaches us that we can not measure position and momentum of a particle with arbitrary certainty. Which still leaves us with a portion of aleatoric uncertainty in our predictions. This type is also called *data uncertainty* [28] because it is inherent to the data itself. Figure 2.2 (a) illustrates this for two overlapping Gaussian distributions. In this case it does not matter how many training examples of the underlying Gaussian distributions we have, examples situated in the overlapping region can not be unambiguously assigned to one of the two classes.

On the other hand, epistemic uncertainty refers to the uncertainty that can be reduced by gathering more information. As a typical machine learning pipeline contains many steps, equally many sources of epistemic uncertainty can be formulated

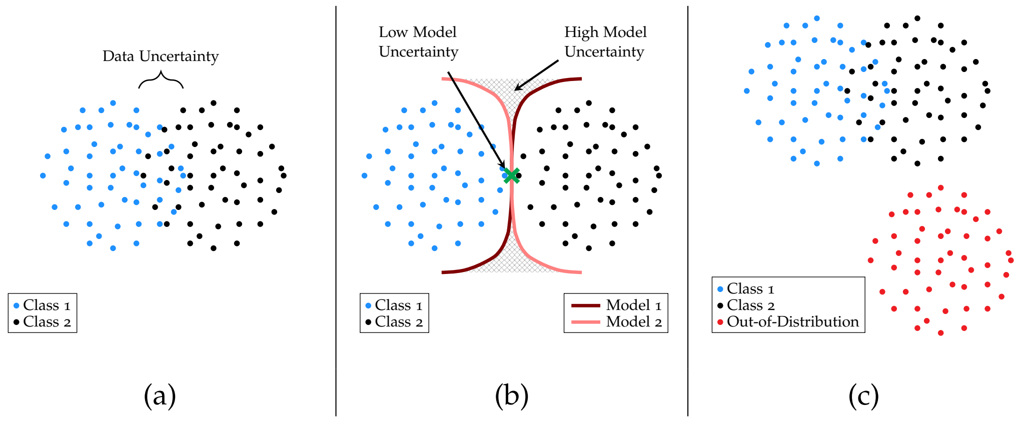

Figure 2.2: Illustration of data (a), model (b), and distributional (c) uncertainty. This graphic is based on Figure 1 from Gawlikowski et al. [38, p. 6].

(see Fig. 2.1). On a broader scale we can distinguish into *model* and *distributional* uncertainty [89].

Model uncertainty summarizes our doubts about whether we specified our model in an optimal way. One source for model uncertainty is the specification of the hypothesis space $\mathcal{H}$ and whether the true risk minimizer h^* is covered by it or not. In statistical learning theory we often assume that $h^* \in \mathcal{H}$, which is called the *realizability assumption* [119, p. 38]. As we will see in Chapter 3, this is actually a realistic assumption for (Deep) Neural Networks. For other algorithms, this uncertainty can be reduced by additional knowledge about the correct (or best) model, which is why it is considered to be of an epistemic nature. Another source of model uncertainty is the choice of model parameters. As mentioned in Section 2.1, the ERM is only an approximation of the true risk minimizer. Depending on the quality and amount of data, the error between h^* and $\hat{h}$ can be larger or smaller, only vanishing in the limit $N \rightarrow \infty$ if our learning algorithm is consistent. This results in an *approximation uncertainty* [119, p. 64, 61, p. 7] and as it can be reduced by having access to more training data, it can also be categorized as being of an epistemic type. There are many more sources of model uncertainty, e.g., the selection of hyperparameters, stochastic decisions like data shuffling, in the case of NNs the model structure and number of parameters [38, pp. 3–4].

Distributional uncertainty is caused by a mismatch between the training distribution and the one that can be observed during the deployment phase. As the world around us is virtually infinite in variety, restricting the learning domain $\mathcal{X}$ as well as the label set $\mathcal{Y}$ is inevitable in order to be able to collect a representative dataset. However, this might result in situations where the model is asked to predict a label for inputs that are not covered by the training set or which can not be mapped to one of the known classes in a meaningful way. This is a problem that typically arises in practical applications and such inputs are called Out-of-Distribution (OoD) as they are outside the known training data distribution. We can expect that every machine learning model in the wild is exposed to these examples, which is why distributional uncertainty always needs to be considered in real world applications. OoD examples are an extreme case of a distribution shift. Often we can observe smaller domain shifts, where the training distribution has a large overlap with the real world data but does not match up completely. This can be self-inflicted, e.g., when the camera which is recording data is calibrated differently compared to the camera that captured the training data, resulting in a shift in the overall color spectrum. Also, it can be induced by external factors like, e.g., in the context of autonomous driving, when the designs of cars change over time.

It is important to notice, which was also pointed out by Hüllermeier and Waegeman [61] and Kiureghian and Ditlevsen [66], that aleatoric and epistemic uncertainty can only be defined without ambiguity if the learning-setting $(\mathcal{X}, \mathcal{Y}, \mathcal{H}, P)$ is fixed.
Consider, e.g., the problem of brain tumor segmentation in Magnetic Resonance Imaging (MRI) images. Due to a limit in MRI image resolution there always remains uncertainty about the boundary of a possible tumor. Even if we would have additional information in form of more scans, the resolution limit still applies, resulting in aleatoric uncertainty at the boundary. Now assume that there is a new medical imaging method that drastically increases the resolution of MRI images. In this case we would be able to reduce the aleatoric uncertainty at the boundary, transferring it to epistemic uncertainty as we now need more data to learn the higher dimensional problem. So was the aleatoric uncertainty experienced with the older imaging technique of epistemic nature in the first place?
In the example above our learning domain $\mathcal{X}$ changed and thus also our learning-setting. Similar transitions between aleatoric and epistemic uncertainty can be observed when allowing a change in the underlying sampling distribution, label or hypothesis space, showing that in order to clearly define aleatoric and epistemic uncertainty, the learning-setting has to be fixed.

If we enable our model to output a distribution over possible outcomes $\hat{p}(y \mid \boldsymbol{x})$, like in Eq. (2.7) (we will see in Chapter 3 that the output of neural networks for classification tasks is also a distribution over classes), we allow it to express aleatoric uncertainty in its predictions. This uncertainty (or lack of knowledge as in Hüllermeier and Waegeman [61, p. 15]) can be measured with the Shannon entropy [120]. For a discrete random variable Y over possible events $y \in \mathcal{Y}$ with the density $p(Y = y) = p_y$, the Shannon entropy is defined as

$$H(p) = -\mathbb{E}_p [\log p] = - \sum_{y \in \mathcal{Y}} p_y \log(p_y) \,. \tag{2.8}$$

Equation (2.8) reaches its minimum of 0 if a single $y \in \mathcal{Y}$ receives all probability mass, and it's maximum of $\log(|\mathcal{Y}|)$ if the probability mass is spread uniformly across all y. We will use the terms *entropy* and *Shannon entropy* interchangeably.

Intuitively the entropy measures a similarity to a uniform distribution over classes. This view of the entropy can also be derived from the Kullback-Leibler Divergence

(KLD) [71]. The KLD between two discrete distributions $p(\boldsymbol{x})$ and $q(\boldsymbol{x})$ is defined as (cf. Bishop [8, p. 55])

$$\mathrm{KL}(p \parallel q) = \mathbb{E}_p \left[\log \left(\frac{p(\boldsymbol{x})}{q(\boldsymbol{x})} \right) \right] = \sum_{\boldsymbol{x}} p(\boldsymbol{x}) \log \left(\frac{p(\boldsymbol{x})}{q(\boldsymbol{x})} \right) . \tag{2.9}$$

Two important properties of the KLD are, that it is not symmetric, meaning $\mathrm{KL}(p \parallel q) \neq \mathrm{KL}(q \parallel p)$, and that it is non-negative $\mathrm{KL}(p \parallel q) \geq 0$ with $\mathrm{KL}(p \parallel q) = 0$ if, and only if $p(\boldsymbol{x}) = q(\boldsymbol{x})$ (see Bishop [8, pp. 55–57] for a proof). The latter results in the KLD being similar to a distance measure between two distributions, although strictly speaking it is not a distance metric because it is not symmetric. The KLD can also be formulated with the cross-entropy and entropy as

$$\mathrm{KL}(p \parallel q) = H(p,q) - H(p) \,, \tag{2.10}$$

with $H(p,q) = -\sum_{\boldsymbol{x}} p(\boldsymbol{x}) \log(q(\boldsymbol{x}))$ being the cross-entropy between p and q. When we now compute the KLD between a class distribution $p(y)$ and the uniform distribution over classes $u(y) = \frac{1}{K}, \forall y \in \mathcal{Y}$ we get

$$\begin{aligned} \mathrm{KL}(p \parallel u) &= \sum_{y \in \mathcal{Y}} p(y) \log \left(\frac{p(y)}{\frac{1}{K}} \right) && (2.11) \\ &= -H(p) + \log(K) \,. && (2.12) \end{aligned}$$

This shows that the (Shannon) entropy is an affine transformation of the KLD with respect to the uniform distribution.

Often the normalized entropy is being used, which results in a quantity similar to a metric with a value range of $[0, 1]$

$$\tilde{H}(p) := -\frac{1}{\log(K)} \sum_{y \in \mathcal{Y}} p_y \log(p_y) \in [0, 1] \,. \tag{2.13}$$

Here, $\tilde{H}(p) = 0$ if the probability mass lies on a single p_y and $\tilde{H}(p) = 1$ if the probability mass is uniformly distributed, therefore $p_y = \frac{1}{K}, \forall y \in \mathcal{Y}$.

When learning point estimates of our model, as we are doing in Eq. (2.4) by only taking the hypothesis which minimizes our empirical risk, we are no longer able to quantify the predictive model uncertainty. Figure 2.2 (b) illustrates this, as we can not determine the model uncertainty without having access to multiple hypothesis. Bayesian theory offers a basis to describe this type of uncertainty by imposing a probabilistic view on the world and describing the model parameters with a distribution

$$p(\boldsymbol{\theta} \mid \mathcal{D}) = \frac{p(\mathcal{D} \mid \boldsymbol{\theta}) p(\boldsymbol{\theta})}{p(\mathcal{D})} \,. \tag{2.14}$$

Here, $\boldsymbol{\theta}$ is a vector of parameters defining a hypothesis, $p(\boldsymbol{\theta} \mid \mathcal{D})$ is the posterior, $p(\boldsymbol{\theta})$ the prior, $p(\mathcal{D} \mid \boldsymbol{\theta})$ the data-likelihood and $p(\mathcal{D}) = \int p(D \mid \boldsymbol{\theta}) p(\boldsymbol{\theta}) \mathrm{d}\boldsymbol{\theta}$ the evidence. Because the posterior is dependent on the data $\mathcal{D}$, we can naturally express the notion of epistemic (model) uncertainty. Gathering more data allows us to describe the true posterior more accurately. If we are able to compute the posterior, we can form predictions by integrating over all parameter configurations

$$p(y \mid \boldsymbol{x}, \mathcal{D}) = \int \underbrace{p(y \mid \boldsymbol{x}, \boldsymbol{\theta})}_{\text{Data}} \underbrace{p(\boldsymbol{\theta} \mid \mathcal{D})}_{\text{Model}} \, \mathrm{d}\boldsymbol{\theta} \,. \tag{2.15}$$

Equation (2.15) can be extended by modeling distributional uncertainty as a distribution over predictive categorical distributions $p(\mu \mid \boldsymbol{x}, \boldsymbol{\theta})$, where μ is a realization of one such distribution [89]. The complete predictive model is then formalized as

$$p(y \mid \boldsymbol{x}, \mathcal{D}) = \int \int \underbrace{p(y \mid \mu)}_{\text{Data}} \underbrace{p(\mu \mid \boldsymbol{x}, \boldsymbol{\theta})}_{\text{Distributional}} \underbrace{p(\boldsymbol{\theta} \mid \mathcal{D})}_{\text{Model}} \, \mathrm{d}\mu \mathrm{d}\boldsymbol{\theta} \,. \tag{2.16}$$

Even though we can model the distributional uncertainty as in Eq. (2.16), most Bayesian approaches do not model it and include it indirectly into the data uncertainty. Depeweg et al. [28] are proposing a way to separate aleatoric and epistemic uncertainty in a Bayesian setup. Their intuition is to quantify the total amount of uncertainty as well as the aleatoric uncertainty. The epistemic part is then the difference between the two. In their work, the total uncertainty is quantified by the entropy over the predictive posterior

$$H(p(y \mid \boldsymbol{x}, \mathcal{D})) = -\sum_{k=1}^{K} p(y = k \mid \boldsymbol{x}, \mathcal{D}) \cdot \log(p(y = k \mid \boldsymbol{x}, \mathcal{D})) \,. \tag{2.17}$$

This quantity is expected to contain the epistemic (model) as well as the aleatoric (data) uncertainty, because due to the marginalization over the parameters the final prediction does no longer depend on them. Similar to the maximum likelihood point estimates, they argue that sampling a single weight configuration from the posterior removes any epistemic uncertainty. The expected entropy over all the parameter realizations should then only contain the aleatoric uncertainty

$$\begin{aligned} \mathbb{E}_{p(\boldsymbol{\theta}|\mathcal{D})} \left[H(p(y \mid \boldsymbol{x}, \boldsymbol{\theta})) \right] = -\int p(\boldsymbol{\theta} \mid \mathcal{D}) \Bigg(\sum_{k=1}^{K} & p(y = k \mid \boldsymbol{x}, \boldsymbol{\theta}) \\ & \cdot \log(p(y = k \mid \boldsymbol{x}, \boldsymbol{\theta})) \Bigg) \mathrm{d}\boldsymbol{\theta} \,. \end{aligned} \tag{2.18}$$

Finally, the epistemic uncertainty is the difference between Eqs. (2.17) and (2.18)

$$I(y, \boldsymbol{\theta}) := H(p(y \mid \boldsymbol{x}, \mathcal{D})) - \mathbb{E}_{p(\boldsymbol{\theta}|\mathcal{D})} \left[H(p(y \mid \boldsymbol{x}, \boldsymbol{\theta})) \right] \,. \tag{2.19}$$

As stated by Depeweg et al. [28], this equals the Mutual Information (MI) between the distribution over classes y and model parameters $\boldsymbol{\theta}$.

A term which is conceptually very similar to uncertainty quantification and often used synonymously throughout the literature is *confidence estimation*. Confidence estimation is the task of assigning a scalar value to the predictions of a neural network which should correspond to the network's confidence in its prediction. As such it should be lower for wrong predictions or unknown inputs and higher for known inputs that are correctly classified. Typically, the value range for a confidence estimate is $[0, 1]$, which makes it suitable to be interpreted as a prediction probability. The main distinction point is that UQ focuses more on quantifying the different sources of uncertainty (aleatoric and epistemic), while a confidence measure summarizes them in a single scalar value. A more apparent difference is that a high confidence should correspond to a correctly classified In-Distribution (ID) example, while the total amount of uncertainty should be small in this case and vice versa. Although confidence estimation is often mixed with uncertainty quantification, confidence can be interpreted in conjunction with uncertainty. Consider, e.g., the problem of weather forecasting, where a model can make a prediction like *"It will rain tomorrow with a probability of* 80 %", in which case our confidence is 80 %. However, it might actually be the case that the current setting is not well represented in our training dataset, imposing a high model and distributional uncertainty on the prediction. As someone who is interpreting the prediction, we would expect a prediction with 80 % confidence to be correct most of the time. More specifically, we would expect it to be correct in exactly 80 % of the cases, which corresponds to a good model calibration. From a practical point of view the complement of the confidence of a well calibrated model (1 − confidence) is already quantifying an aggregation of data and distributional uncertainty. Therefore, we will see confidence estimation as a type of uncertainty quantification and are using the two terms interchangeably.

2.3 CONTROLLING SHIFTS IN LEARNING-SETTINGS

Assuming our learning-setting $(\mathcal{X}, \mathcal{Y}, \mathcal{H}, P)$ to be fixed is helpful when arguing about the exact type of uncertainty. However, when deploying machine learning models in the real world, this assumption can not be considered fulfilled.

As an example, consider the problem of semantic segmentation of street scenes for automated driving. Domain shifts on $\mathcal{X}$ can be caused by changes in location (e.g., Asian vs. European cities or city vs. countryside), short time differences (e.g., day $\leftrightarrow$ night or winter $\leftrightarrow$ summer) or longer time differences (e.g., the designs of cars change over durations of decades). Similarly, $\mathcal{Y}$ might change over time due to, e.g., newly arising vehicle classes that were not present during the initial recording of the training dataset. The frequency of pedestrians in these street scenes might change due to an increasing population size, effectively changing the underlying sampling distribution $P(X, Y)$. Additionally, the visual world is very high-dimensional, making it practically impossible to craft a dataset that is covering everything, thus making it inevitable to enforce a restriction to a subspace for which a representative dataset can be created.

The differentiation into aleatoric and epistemic uncertainty has a low relevance for ad-hoc predictions. The exact source of the uncertainty in, for example, the

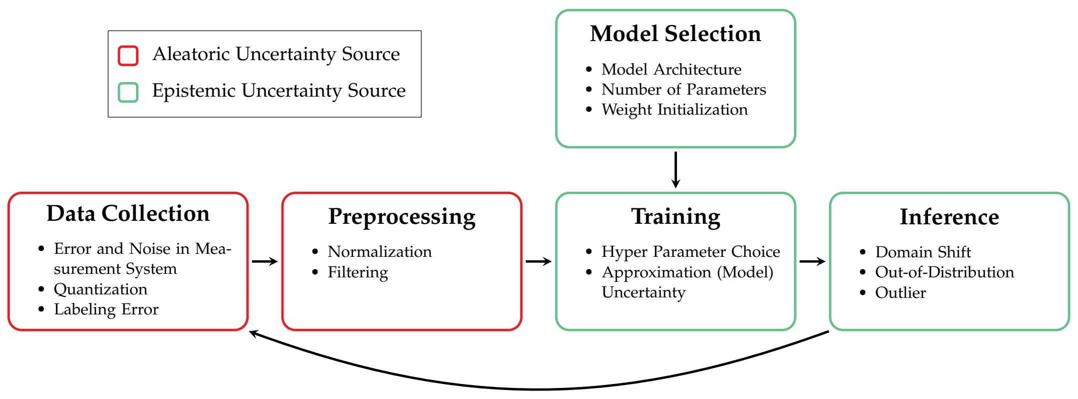

Figure 2.3: Feedback loop added to Fig. 2.1. Unknown inputs and domain shifts should be detected during inference and trigger the collection of additional data. Uncertainty measures can trigger the loop as well as give advice on which data to collect.

semantic segmentation map during driving is of secondary interest. Whether the ML system can not make reliable predictions because it has never seen parts of the scene or encountered an ambiguous input is irrelevant in this situation. In any case, the driver should be requested to take over or the vehicle should be stopped immediately. As a consequence, quantifying the total amount of uncertainty can be sufficient in this case. However, the distinction becomes valuable when trying to control shifts in the underlying learning-setting. Controlling these kinds of shifts can only be accomplished by adding a loop into the ML development pipeline, as can be seen in Fig. 2.3. Monitoring the level of aleatoric and epistemic uncertainty during the deployment phase makes it possible to trigger the start of this loop. Samples attributed to high aleatoric or epistemic uncertainty can be collected and, depending on the uncertainty source, be fed to different tooling steps. Assuming our training dataset is representative for the learning-setting, new classes, which occur during the deployment phase, can be detected by monitoring the model's epistemic uncertainty. When the overall level of epistemic uncertainty is high for many inputs, a human can decide to add some of these new classes to the model. The detected instances can then be annotated (e.g., by using active learning) or used in a semi-supervised learning framework and fed back into the training dataset. Similarly, we can use the model's aleatoric uncertainty to decide if better sensors are needed or to detect errors in the data acquisition pipeline.

A framework to efficiently identify unknown samples was proposed by Oberdiek, Rottmann, and Fink [99]. The workflow was demonstrated on the task of semantic segmentation of street scenes. Segments with a low predicted Intersection over Union (IoU) to the annotation, as produced by the MetaSeg [108] approach, are collected and projected into a low dimensional feature representation using a pre-trained CNN (see Section 3.5). Using distance metrics such as the *cosine similarity* or *Euclidean Norm*, segments which are similar to a user defined example query, are retrieved. Semantically similar objects are close to one another in the feature space, enabling the user to identify groups of objects with a high occurrence frequency.

Clustering techniques are used to support the identification of new object categories. By utilizing such a retrieval based approach, many samples can be searched fast for unknown classes. In further downstream tasks, the identified segments might then be labeled and added to the training dataset. This decreases the epistemic uncertainty caused by the respective objects for a model which is trained on the extended dataset.

3 NEURAL NETWORKS

This chapter will introduce the theory of DNNs. After a brief review of the beginnings of NNs, the backpropagation algorithm is derived using an MLP in Section 3.3. In the same section we gain insights into connections between Maximum Likelihood Estimation (MLE) and the training of NNs, which aligns with the computation of uncertainty from probability distributions, as discussed in Chapter 2. Afterwards, the concept of CNNs and basic building blocks of modern DNNs are introduced in Section 3.5. Following this, a short overview of the history of DNN architectures and a more detailed description of the models used in the experiments is given. Additionally, there will be comments whenever an algorithm or architectural choice of NNs is causing uncertainty.

3.1 PERCEPTRON

A *Perceptron*, as proposed by Rosenblatt [107], computes an affine transformation of its inputs by multiplying each with a learnable weight and adding a bias term. It was first invented by McCulloch and Pitts [90] who formulate it with binary inputs and outputs exclusively. Rosenblatt [107] extended this theory by incorporating the learnable weighting and the bias term. The bias added to the weighted sum of inputs (also called activation) is then transformed by applying a non-linear activation function. A simple illustration of this can be found in Fig. 3.1 (b). Mathematically a Perceptron can be formulated as a function $f : \mathbb{R}^{d_{\boldsymbol{x}}} \to \mathbb{R}$ with weights $\boldsymbol{w} \in \mathbb{R}^{d_{\boldsymbol{x}}}$, a bias term $b \in \mathbb{R}$, a non-linear activation function $\phi : \mathbb{R} \to \mathbb{R}$ and an input $\boldsymbol{x} \in \mathbb{R}^{d_{\boldsymbol{x}}}$

$$f(\boldsymbol{x} \mid \boldsymbol{w}) = \phi\left(\left(\sum_{i=1}^{n} \boldsymbol{w}_i \boldsymbol{x}_i\right) + b\right) . \tag{3.1}$$

In the original formulation of Rosenblatt [107], ϕ is chosen as a step function

$$\phi(a) = \begin{cases} 1, & \text{if } a > 0 \\ 0, & \textit{else} \end{cases} . \tag{3.2}$$

This model then naturally allows performing binary classification without applying further decision rules. In *Perceptrons: An Introduction to Computational Geometry*, Minsky and Papert [92] showed limitations of the Perceptron, especially its inability to solve non-linear problems, e.g., the XOR-Problem.

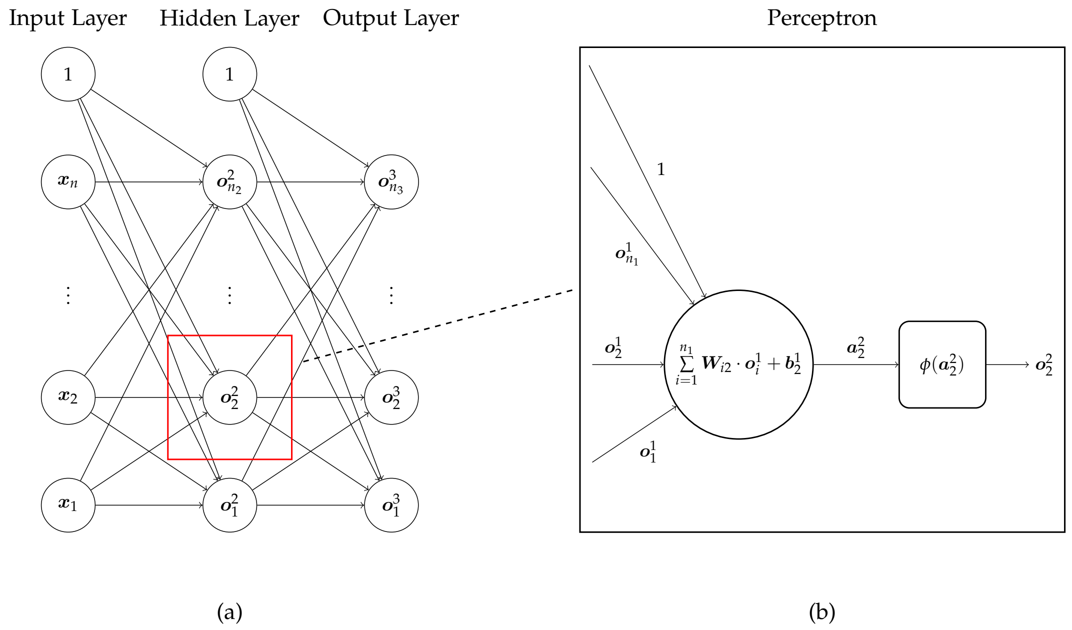

Figure 3.1: Illustration of (a) a Multilayer Perceptron (MLP) with one hidden layer and (b) a Perceptron.

3.2 MULTILAYER PERCEPTRON

As mentioned in Section 3.1, Minsky and Papert [92] demonstrated the inability of a single Perceptron to model the XOR-function. However, they also showed that stacking multiple layers of Perceptrons can alleviate this limitation. This architecture of stacked Perceptrons is today called a *Multilayer Perceptron* (MLP) or *Feed Forward Neural Network*. The latter name originates from the fact that in this type of architecture, neurons are only connected "forward" from the previous and to the next layer, and thus the model can be viewed as a directed graph with no cycles. In this section we will take a look at this very basic model of an NN.

We define a single layer in an MLP, having index $l \in \{1, \dots, L-1\}$, as a nonlinear function $\boldsymbol{f}^l : \mathbb{R}^{n_l} \to \mathbb{R}^{n_{l+1}}$ with n_l input neurons, n_{l+1} output neurons, a weight matrix $\boldsymbol{W} \in \mathbb{R}^{n_l \times n_{l+1}}$, bias weights $\boldsymbol{b} \in \mathbb{R}^{n_{l+1}}$, non-linear activation functions ϕ^l and an input $\boldsymbol{z} \in \mathbb{R}^{n_l}$

$$\boldsymbol{f}^l(\boldsymbol{z} \mid \boldsymbol{W}) = (\boldsymbol{f}^l(\boldsymbol{z} \mid \boldsymbol{W})_1, \dots, \boldsymbol{f}^l(\boldsymbol{z} \mid \boldsymbol{W})_{n_{l+1}})^\intercal , \tag{3.3}$$

$$\boldsymbol{f}^l(\boldsymbol{z} \mid \boldsymbol{W})_j = \phi^{l+1}\left(\left(\sum_{i=1}^{n_l} \boldsymbol{W}_{ij}\boldsymbol{z}_i\right) + \boldsymbol{b}_j\right) = \phi^{l+1}\left(\boldsymbol{W}_{\cdot j}^\intercal \boldsymbol{z} + \boldsymbol{b}_j\right) , \tag{3.4}$$

$$\forall\, j = 1, \dots, n_{l+1}, \quad l = 1, \dots, L-1$$

The complete MLP can then be defined by chaining together all individual layers, with $\boldsymbol{\theta}$ representing the vector of all model weights and $\boldsymbol{x}$ the input to the first layer

$$\boldsymbol{f}(\boldsymbol{x} \mid \boldsymbol{\theta}) = (\boldsymbol{f}^{L-1} \circ \ldots \circ \boldsymbol{f}^{1})(\boldsymbol{x} \mid \boldsymbol{\theta}) . \tag{3.5}$$

In order for an NN to be able to solve non-linear problems a non-linear activation function is indispensable. If all ϕ^l were linear, the whole network would collapse to a single affine transformation, making it impossible to even solve the XOR problem. In the early days of NN research the sigmoid function was used as activation function

$$\sigma(a) = \frac{1}{1+\mathrm{e}^{-a}}, \, a \in \mathbb{R} . \tag{3.6}$$

The nice property about this function is that it is differentiable everywhere with

$$\sigma'(a) = \sigma(x)(1-\sigma(a)) . \tag{3.7}$$

This derivative reaches its maximum at $a = 0$ with $\sigma(0) = 0.25$ and tends towards 0 for $a \to \pm\infty$. In Section 3.3 we will see implications of this value range on the training process. In Fig. 3.1 you can find a simple illustration of an MLP with one hidden layer.

It has been proven by Hornik, Stinchcombe, and White [56] that a 3-layer MLP (input, output and one hidden-layer) has a universal approximation property, given that the hidden layer has a sufficient number of neurons (approaching infinity). Although the result could not yet be generalized to arbitrary NN architectures, it makes the *realizability assumption* from Section 2.2 fairly realistic. This means that the model uncertainty resulting from the (miss-) specification of the hypothesis space $\mathcal{H}$ can be neglected to a large extent, when using DNNs.

3.3 TRAINING

NNs are usually trained in a supervised learning framework. As explained in Section 2.1, our training dataset consists of N i.i.d. data points $\mathcal{S} = \{(\boldsymbol{x}^{(1)}, y^{(1)}), \ldots, (\boldsymbol{x}^{(N)}, y^{(N)})\}$, having features $\boldsymbol{x} \in \mathcal{X}$ and labels $y \in \mathcal{Y} = \{1, \ldots, K\}$, which were drawn from the underlying joint sampling distribution $p(\boldsymbol{x}, y)$. Categorical values as in $\mathcal{Y}$ are typically represented using a *one-hot* encoding

$$\boldsymbol{y}_i^{\mathrm{oh}} = \mathbb{1}_{\{i=y\}} . \tag{3.8}$$

Similarly, we model the output layer of an NN with K output neurons, where each neuron represents one of the classes in $\mathcal{Y}$. On top of that, applying the softmax activation function

$$\begin{aligned} \hat{\boldsymbol{y}}_i &:= \hat{p}(y = i \mid \boldsymbol{x}, \boldsymbol{\theta}) = \frac{\mathrm{e}^{\boldsymbol{f}(\boldsymbol{x}|\boldsymbol{\theta})_i}}{\sum_{j=1}^{K} \mathrm{e}^{\boldsymbol{f}(\boldsymbol{x}|\boldsymbol{\theta})_j}}, \quad \forall i = 1, \ldots, K, \\ \sum_{i=1}^{K} \hat{\boldsymbol{y}}_i &= 1 , \end{aligned} \tag{3.9}$$

induces a distribution over classes $\hat{p}(y \mid \boldsymbol{x}, \boldsymbol{\theta}) \in [0,1]^K$ in the output of the neural network. For brevity, we will sometimes leave out the $\boldsymbol{\theta}$ and write $\hat{p}(y \mid \boldsymbol{x})$ for the class distribution predicted by the NN. The outputs of the second to last layer (also called logits) will be denoted by $\boldsymbol{f}(\boldsymbol{x} \mid \boldsymbol{\theta})$.

3.3.1 *Loss Functions*

Our goal is to adjust the weights $\boldsymbol{\theta}$ of the NN in such a way, that we minimize a loss function ℓ (risk as in Section 2.1) between the network's prediction $\hat{\boldsymbol{y}} = \hat{p}(y \mid \boldsymbol{x}, \boldsymbol{\theta})$ and the label y. The choice of ℓ depends on the type of problem at hand. For now, let us choose ℓ as the squared L_2 loss between $\hat{\boldsymbol{y}}$ and $\boldsymbol{y}^{\text{oh}}$, also commonly used in least squares estimation for linear regression problems, and define $\mathfrak{L}$ as the mean squared loss

$$\ell(\hat{\boldsymbol{y}}, \boldsymbol{y}^{\text{oh}}) = \|\hat{\boldsymbol{y}} - \boldsymbol{y}^{\text{oh}}\|_2^2, \tag{3.10}$$

$$\mathfrak{L}(\boldsymbol{\theta}, \mathcal{S}) = \frac{1}{|\mathcal{S}|} \sum_{(\boldsymbol{x},y)\in\mathcal{S}} \ell(\hat{\boldsymbol{y}}, \boldsymbol{y}^{\text{oh}}) \,. \tag{3.11}$$

By minimizing $\mathfrak{L}$ w.r.t. $\boldsymbol{\theta}$ we are obtaining a set of model parameters

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \mathfrak{L}(\boldsymbol{\theta}, \mathcal{S}) \,. \tag{3.12}$$

If $\hat{p}$ is linear, as for linear regression problems, this choice of loss function results in a convex optimization problem with a global minimum which can be found analytically. For NNs however, $\hat{p}$ is in general too complex, resulting in many local minima in $\mathfrak{L}$. This makes it practically impossible to find a global minimum analytically. Consequently, in order to train NNs, we will need to utilize a different approach, which will be introduced later in this section.

In the following, we will see why it is important to choose the loss function in relation to the problem to be solved. For this we consider the training of neural networks in the context of *Maximum Likelihood Estimation* (MLE), which is a method for computing point estimates of parameterized models. The intuitive idea behind MLE is to find a set of parameters that maximizes the model likelihood of observing the data points $(\boldsymbol{x}, y) \in \mathcal{S}$. A crucial assumption to be able to compute the likelihood of our sample $\mathcal{S}$ is the independence of our data points, which is why we required our data points to be i.i.d., as previously discussed. The minimization of the mean squared loss function can be rewritten as

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \frac{1}{|\mathcal{S}|} \sum_{(\boldsymbol{x},y)\in\mathcal{S}} \|\hat{\boldsymbol{y}} - \boldsymbol{y}^{\text{oh}}\|_2^2 \tag{3.13}$$

$$= \arg\min_{\boldsymbol{\theta}} \sum_{(\boldsymbol{x},y)\in\mathcal{S}} \left[\frac{K}{2}\log(2\pi) + \frac{1}{2}\|\hat{\boldsymbol{y}} - \boldsymbol{y}^{\text{oh}}\|_2^2\right] \tag{3.14}$$

$$= \arg\max_{\boldsymbol{\theta}} \sum_{(\boldsymbol{x},y)\in\mathcal{S}} \log\left(\frac{1}{\sqrt{(2\pi)^K}}\,\mathrm{e}^{-\frac{1}{2}(\hat{\boldsymbol{y}}-\boldsymbol{y}^{\text{oh}})^\intercal(\hat{\boldsymbol{y}}-\boldsymbol{y}^{\text{oh}})}\right) \tag{3.15}$$

$$= \arg\max_{\boldsymbol{\theta}} \sum_{(\boldsymbol{x},y)\in\mathcal{S}} \log\left(\mathcal{N}_K(\hat{\boldsymbol{y}} \mid \boldsymbol{y}^{\text{oh}}, \mathbb{I}_K)\right) \tag{3.16}$$

$$= \arg\max_{\boldsymbol{\theta}} \prod_{(\boldsymbol{x},y)\in\mathcal{S}} \mathcal{N}_K(\hat{\boldsymbol{y}} \mid \boldsymbol{y}^{\text{oh}}, \mathbb{I}_K)\,. \tag{3.17}$$

In Eq. (3.14) we made a smart extension with the constant $\frac{K}{2}\log(2\pi)$, which later serves as the first factor in the Gaussian density. From Eq. (3.14) to Eq. (3.15) we multiplied the equation by -1, which changed the optimization goal from arg min to arg max. Afterwards, a $\log(\exp(\cdot))$ expansion on the quadratic L_2-norm allowed to merge the equation into a single logarithm. Equation (3.17) (without the $\arg\max_{\boldsymbol{\theta}}$) is the likelihood of observing all $(\boldsymbol{x}, y) \in \mathcal{S}$ when making a (multivariate) Gaussian model assumption on the data, having unit covariance $\mathbb{I}_K$ and the annotation $\boldsymbol{y}^{\text{oh}}$ as mean. In this assumption we are assuming a constant observation noise of 1 for each class, effectively ignoring any uncertainty originating from the collected data. This shows that the minimization of our mean squared loss function is actually equivalent to MLE. Considering the Gaussian distributional assumption on the target data, the mean squared loss function is a reasonable choice for problems in which the data is following a Gaussian distribution. If we are trying to solve a regression problem, this is a plausible choice, as the data points are real-valued and not box-constrained like a categorical distribution. However, in the case of a classification problem, considering the data to follow a normal distribution is no longer reasonable as these are discrete assignment problems.

Therefore, it makes sense to model the data as a *categorical distribution*, which is a generalization of the *Bernoulli distribution* to a categorical random variable, and results in the data model assumption

$$p(y \mid \boldsymbol{x}) = B(y, p_1, \dots, p_K) = \prod_{i=1}^{K} p_i^{\mathbb{1}_{\{y=i\}}}\,. \tag{3.18}$$

This allows us to write the maximization of the likelihood as

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \prod_{(\boldsymbol{x},y)\in\mathcal{S}} B(y, \hat{\boldsymbol{y}}_1, \ldots, \hat{\boldsymbol{y}}_K) \tag{3.19}$$

$$= \arg\max_{\boldsymbol{\theta}} \prod_{(\boldsymbol{x},y)\in\mathcal{S}} \prod_{i=1}^{K} \hat{\boldsymbol{y}}_i^{\boldsymbol{y}_i^{\mathrm{oh}}} \tag{3.20}$$

$$= \arg\max_{\boldsymbol{\theta}} \prod_{(\boldsymbol{x},y)\in\mathcal{S}} \hat{\boldsymbol{y}}_y \tag{3.21}$$

$$= \arg\max_{\boldsymbol{\theta}} \sum_{(\boldsymbol{x},y)\in\mathcal{S}} \log(\hat{\boldsymbol{y}}_y) \tag{3.22}$$

$$= \arg\min_{\boldsymbol{\theta}} \frac{1}{|\mathcal{S}|} \sum_{(\boldsymbol{x},y)\in\mathcal{S}} -\log(\hat{\boldsymbol{y}}_y)\,. \tag{3.23}$$

Equation (3.23) (without the $\arg\min_{\boldsymbol{\theta}}$) is also called the *Negative Log-Likelihood Loss* (NLLL) or *Cross-Entropy-Loss* and is the standard for classification problems.

To finally form a discrete prediction from the predicted class distribution $\hat{p}$, a common approach is to use the Maximum a Posteriori (MAP) or *Bayes* decision rule

$$\hat{y}^{\mathrm{MAP}} := \arg\max_{i=1,\ldots,K} \hat{\boldsymbol{y}}_i\,. \tag{3.24}$$

Because we are choosing a class based on the maximum posterior probability, we are discarding any sense of aleatoric uncertainty that might be caused by other classes. But as described in Section 2.2 we can measure the aleatoric uncertainty before applying the MAP decision rule by computing the Shannon entropy over the predicted class distribution

$$\tilde{H}(\hat{\boldsymbol{y}}) = -\frac{1}{\log(K)} \sum_{i=1}^{K} \hat{\boldsymbol{y}}_i \log(\hat{\boldsymbol{y}}_i)\;. \tag{3.25}$$

Another interesting observation is, that the cross-entropy loss function naturally arises by computing the KLD between the annotation $\boldsymbol{y}^{\mathrm{oh}}$ and the predicted class distribution $\hat{\boldsymbol{y}}$

$$\mathrm{KL}(\boldsymbol{y}^{\mathrm{oh}} \parallel \hat{\boldsymbol{y}}) = \sum_{i=1}^{K} \boldsymbol{y}_i^{\mathrm{oh}} \log\left(\frac{\boldsymbol{y}_i^{\mathrm{oh}}}{\hat{\boldsymbol{y}}_i}\right) \tag{3.26}$$

$$= -\log(\hat{\boldsymbol{y}}_y) + \underbrace{H(\boldsymbol{y}^{\mathrm{oh}})}_{=0}\;. \tag{3.27}$$

Here we set $0 \cdot \log(0) = 0$, which is consistent with the limit. This also connects the Shannon entropy in a principled way to the cross-entropy loss, as discussed in Section 2.2.

3.3.2 *Gradient Descent*

As mentioned above, finding an analytic solution to Eq. (3.17) or Eq. (3.23) is usually not possible as $\hat{p}$ is too complex. However, if all involved functions in the computation of $\mathfrak{L}$ are differentiable, we can use the *gradient descent* optimization algorithm. By iteratively making steps in the direction of the steepest descent, the algorithm is guaranteed to converge to a local minimum [22], assuming some intelligent selection of the step size. If the function to optimize over is convex, the algorithm will also converge to the global minimum. The update rule can be defined as

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \cdot \nabla_{\boldsymbol{\theta}} \mathfrak{L} \,. \tag{3.28}$$

Here $\eta \in \mathbb{R}_{>0}$ is the learning rate which is a hyperparameter of the algorithm. From a practical point of view, depending on the dataset size, computing the loss over all samples can be very costly and results in a slow convergence speed. This is the reason why the Stochastic Gradient Descent (SGD) [10] variant was developed. In its extreme form there happens a weight update after each single sample drawn from the training set S. This improves the convergence speed but also introduces a lot of variance in the optimization process. Today, the *batch stochastic gradient descent* is predominantly used for optimizing NNs. This variant uses a batch of data points $\mathcal{S}_B \subseteq \mathcal{S}$ in each weight update, where usually $|\mathcal{S}_B| \ll |\mathcal{S}|$. Small batches have the advantage of fast loss computation and, when uniformly randomly sampled from S, approximate the loss over the whole dataset. Many extensions to and modifications of the SGD exist today which try to improve the convergence speed (e.g., SGD with momentum [127], AdaGrad [33], RMSProp [131], Adam [64]). Although *batch stochastic gradient descent* is used in practice, we will stick to the entire training set $\mathcal{S}$ in the theoretical derivations for ease of notation, unless otherwise noted.

Training NNs and especially DNNs has been a long-standing problem in the field. The introduction of the *backpropagation algorithm* [110] was one of the key advancements to enable efficient training of these models. The following section will give an introduction to this algorithm with an application to MLPs. It can however be extended to more complex architectures using, e.g., convolutions.

Generally speaking the algorithm consists of four steps:

1. **Forward pass:** An input $\boldsymbol{x}$ (or rather a whole batch of inputs as in the *batch stochastic gradient descent*) is presented to the network, and its output alongside with all intermediate neuron activations are computed.

2. **Loss computation:** Based on the output $\hat{\boldsymbol{y}}$ of the NN and the label y the average error of the prediction is computed by using the loss function ℓ.

3. **Backward pass:** Compute the gradients of the loss w.r.t. each weight by backpropagating the error from layer to layer.

4. **Weight adjustment:** Update each weight according to Eq. (3.28) given the gradient.

The name *backpropagation* comes from the fact that the contribution to the loss function of each neuron can be computed given the errors of the neurons in the following layer. Thus, the error is propagated back from layer to layer, starting at the output. As shown below, the formulas for the algorithm can be derived by applying the chain rule for differentiation.

Consider an MLP as defined in Section 3.2. We define $\boldsymbol{a}_j^l$ as the activation of neuron j in layer l, $\boldsymbol{o}_j^l$ as the output of the neuron j in layer l, $\phi^l : \mathbb{R} \to \mathbb{R}$ the activation function for neurons in layer l, $\boldsymbol{W}_{ij}^l$ the weight between neuron i in layer l and neuron j in layer $l+1$, $\boldsymbol{b}_j^l$ the bias weight of neuron j in layer l and $\mathfrak{L}$ as an arbitrary differentiable loss computed on the training dataset $\mathcal{S}$.

For $1 \le l \le L-1$ the gradients are computed as

$$\frac{\partial \mathfrak{L}}{\partial \boldsymbol{W}_{ij}^l} = \frac{\partial \mathfrak{L}}{\partial \boldsymbol{o}_j^{l+1}} \cdot \frac{\partial \boldsymbol{o}_j^{l+1}}{\partial \boldsymbol{a}_j^{l+1}} \cdot \frac{\partial \boldsymbol{a}_j^{l+1}}{\partial \boldsymbol{W}_{ij}^l} \tag{3.29}$$

$$= \underbrace{\frac{\partial \mathfrak{L}}{\partial \boldsymbol{o}_j^{l+1}} \cdot \phi^{l+1'}\left(\boldsymbol{a}_j^{l+1}\right)}_{\boldsymbol{\delta}_j^{l+1}} \cdot \boldsymbol{o}_i^l \tag{3.30}$$

$$= \boldsymbol{\delta}_j^{l+1} \cdot \boldsymbol{o}_i^l, \quad \forall i = 1, \dots, n_l, j = 1, \dots, n_{l+1}, \tag{3.31}$$

$$\frac{\mathfrak{L}}{\partial \boldsymbol{b}_j^l} = \frac{\partial \mathfrak{L}}{\partial \boldsymbol{o}_j^{l+1}} \cdot \frac{\partial \boldsymbol{o}_j^{l+1}}{\partial \boldsymbol{a}_j^{l+1}} \cdot \frac{\partial \boldsymbol{a}_j^{l+1}}{\partial \boldsymbol{b}_j^l} \tag{3.32}$$

$$= \underbrace{\frac{\partial \mathfrak{L}}{\partial \boldsymbol{o}_j^{l+1}} \cdot \phi^{l+1'}\left(\boldsymbol{a}_j^{l+1}\right)}_{\boldsymbol{\delta}_j^{l+1}} \tag{3.33}$$

$$= \boldsymbol{\delta}_j^{l+1}, \quad \forall j = 1, \dots, n_{l+1}. \tag{3.34}$$

$$\tag{3.35}$$

Given the definition of $\boldsymbol{\delta}_j^l$ from above, we can follow

$$\boldsymbol{\delta}_j^l = \phi^{l'}\left(\boldsymbol{a}_j^l\right) \cdot \frac{\partial \mathfrak{L}}{\partial \boldsymbol{o}_j^l} \tag{3.36}$$

$$= \phi^{l'}\left(\boldsymbol{a}_j^l\right) \cdot \sum_{k=1}^{n_{l+1}} \left(\frac{\partial \mathfrak{L}}{\partial \boldsymbol{o}_k^{l+1}} \cdot \frac{\partial \boldsymbol{o}_k^{l+1}}{\partial \boldsymbol{a}_k^{l+1}} \cdot \frac{\partial \boldsymbol{a}_k^{l+1}}{\partial \boldsymbol{o}_j^l} \right) \tag{3.37}$$

$$= \phi^{l'}\left(\boldsymbol{a}_j^l\right) \cdot \sum_{k=1}^{n_{l+1}} \left(\underbrace{\frac{\partial \mathfrak{L}}{\partial \boldsymbol{o}_k^{l+1}} \cdot \phi^{l+1'}\left(\boldsymbol{a}_k^{l+1}\right)}_{\boldsymbol{\delta}_k^{l+1}} \cdot \boldsymbol{W}_{jk}^l \right) \tag{3.38}$$

$$= \phi^{l'}\left(\boldsymbol{a}_j^l\right) \cdot \sum_{k=1}^{n_{l+1}} \left(\boldsymbol{\delta}_k^{l+1} \cdot \boldsymbol{W}_{jk}^{l+1} \right), \quad \forall j = 1, \dots, n_l. \tag{3.39}$$

In Eq. (3.37) we rewrote the derivative w.r.t. $\boldsymbol{o}_j^l$ as a sum of the derivatives w.r.t. the output of the following layer by applying the chain-rule. As the last layer L does not have following layers from which the loss can be backpropagated we need to define $\boldsymbol{\delta}_j^l$ as

$$\boldsymbol{\delta}_j^l = \begin{cases} \frac{\partial \mathfrak{L}}{\partial \boldsymbol{o}_j^l} \cdot \phi^{l'}(\boldsymbol{a}_j^l), & l = L \\ \phi^{l'}\left(\boldsymbol{a}_j^l\right) \cdot \sum_{k=1}^{n_{l+1}} \left(\boldsymbol{\delta}_k^{l+1} \cdot \boldsymbol{W}_{jk}^l\right), & 1 \leq l \leq L-1 \end{cases} . \tag{3.40}$$

Considering the derivations from above, Eq. (3.28) for an MLP can be formulated as

$$\boldsymbol{W}_{ij}^l \leftarrow \boldsymbol{W}_{ij}^l - \eta \cdot \frac{\partial \mathfrak{L}}{\partial \boldsymbol{W}_{ij}^l} = \boldsymbol{W}_{ij}^l - \eta \cdot \boldsymbol{\delta}_j^{l+1} \cdot \boldsymbol{o}_i^l \qquad \begin{array}{l} i = 1, \dots, n_l \\ j = 1, \dots, n_{l+1} \\ l = 1, \dots, L-1 \end{array} , \tag{3.41}$$

$$\boldsymbol{b}_j^l \leftarrow \boldsymbol{b}_j^l - \eta \frac{\partial \mathfrak{L}}{\partial \boldsymbol{b}_j^l} = \boldsymbol{b}_j^l - \eta \cdot \boldsymbol{\delta}_j^{l+1} \qquad \begin{array}{l} j = 1, \dots, n_{l+1} \\ l = 1, \dots, L-1 \end{array} . \tag{3.42}$$

3.3.3 *Uncertainty resulting from gradient descent optimization*

Although the backpropagation algorithm is the main driver behind today's advances in neural network research, it also introduces a lot of model uncertainty. We saw previously in this section that the *realizability assumption* can be considered fulfilled for the class of DNNs. But as described in Section 2.1, another source of model uncertainty results from the *approximation* $\hat{h}$ of the true risk minimizer h^*. If our sample size goes to infinity ($|\mathcal{S}| \to \infty$), we should be able to compute the true risk minimizer. In practice however, multiple sources prevent us from reducing the approximation uncertainty to zero. First, our dataset size will always be finite and second, the (batch stochastic) gradient descent algorithm can not guarantee that we reach the global minimum as the loss function used for training a neural network is in general not convex. Thus, the learning will most likely converge to a local minimum within the loss landscape. It has been demonstrated that changes in the architecture of the neural network can make the optimization landscape smoother [82] but in general the loss function has many local minima which also will change depending on the setting of a multitude of different hyperparameters and random effects during the training (see [38, p. 4]), such as

- Architecture of the neural network
- Weight initialization
- Random sampling of the data batches
- Batch size
- Learning rate

In total, this results in a large portion of epistemic uncertainty in our predictions, as we can never be sure that we found the true risk minimizer. Methods such as ensembling [75] are making use of the fact that every training run of a neural network will result in a different local minimum and build a group of diverse experts. These ensembles of experts not only increase the final classification accuracy but are also able to express confidence and uncertainty in a more calibrated way (more on that in Section 4.3).

As an intermediate conclusion one can say that methods for quantifying the uncertainty of DNN predictions should definitely be able to quantify model uncertainty as this makes up a large part of the epistemic uncertainty.

3.3.4 *Vanishing Gradients and Dead Neurons*

As mentioned in Section 3.2, in the beginning of NN research the sigmoid activation function was used for hidden layers, thus $\phi^l(x) = \sigma(x)\,, \forall l = 1, \dots, L-1$. Now that we know how NNs are optimized we can also understand a problem with this choice of activation function. Assume we have an MLP with one neuron in each layer, thus $n_l = 1\,, \forall l = 1, \dots, L$, and the sigmoid activation function for hidden layers. For simpler notation also assume, without loss of generality, that all weights are set to 1. If we now compute the gradient w.r.t. $\boldsymbol{W}^1_{11}$ we get

$$\begin{aligned}
\frac{\partial \mathfrak{L}}{\partial \boldsymbol{W}^1_{11}} &= o^1_1 \cdot \delta^2_1 \\
&= o^1_1 \cdot \sigma'\left(a^2_1\right) \cdot \delta^3_1 \\
&\vdots \\
&= o^1_1 \prod_{l=2}^{L} \sigma'\left(a^l_1\right) \\
&\leq o^1_1 \cdot \big(\max_{l=2,\dots,L} \sigma'(a^l_1)\big)^{L-1} \\
&= o^1_1 \cdot 0.25^{L-1}
\end{aligned} \tag{3.43}$$

The inequality in Eq. (3.43) is especially harmful for DNNs (large L) as the gradient tends towards zero, resulting in stagnant weight updates for weights which are closer to the input layer. This problem is called *vanishing gradient problem* in the literature [41, p. 290]. By choosing a different activation function called Rectified Linear Unit (ReLU), we can circumvent the problem to a large extent

$$\mathrm{ReLU}(x) = \begin{cases} x, & x > 0 \\ 0, & x \leq 0 \end{cases}. \tag{3.44}$$

Strictly speaking, the ReLU activation function is not differentiable at $x = 0$. In a practical scenario this is not a problem as one can define the derivative as

$$\mathrm{ReLU}'(x) = \begin{cases} 1, & x > 0 \\ 0, & x \leq 0 \end{cases}. \tag{3.45}$$

However, the ReLU can lead to another optimization problem, which is the one of *dead neurons* [41, p. 238]. Dead neurons occur if the input to the neuron is negative, resulting in a zero gradient. The problem is even worse if in some point of the training process a large weight update happens, producing a negative weight and bias. In this case, as outputs of previous layers are always positive with a ReLU, the activation is always negative, preventing the update of the respective weights for the rest of the training. To avoid this, a small modification to the ReLU suffices which is then called Leaky Rectified Linear Unit (LeakyReLU)

$$\mathrm{LeakyReLU}(x) = \begin{cases} x, & x > 0 \\ \lambda x, & x \leq 0 \end{cases}, \quad \lambda \in \mathbb{R}_{>0}. \tag{3.46}$$

Usually one chooses $\lambda \in (0, 1)$ to allow a small gradient to pass through from which the addition *Leaky* originates. Note that choosing $\lambda = 1$ should be avoided as this results in collapse of the neural network to an affine transformation (see Section 3.2). Glorot, Bordes, and Bengio [40] argue that sparse networks resulting from individual dead neurons do not constitute a significant problem in the training of neural networks. Although they show experimental results supporting this claim, the authors only experiment with very shallow networks.

Analyzing the backpropagation algorithm, especially Eq. (3.40), we can see that the gradient of a parameter is not only dependent on ϕ' but also on the weights to the subsequent layer. In the case of small weights, some parameters still might only receive a small gradient. This is especially apparent when applying regularization techniques such as L_2 regularization (see Section 3.4), where the weights are optimized towards a zero mean multivariate Gaussian.

3.3.5 *Implications of piecewise affine activation functions*

Piecewise affine activation functions such as ReLU or LeakyReLU are solving the problems of *vanishing gradients* and *dead neurons*. They do however introduce other difficulties with respect to UQ of NNs. As shown by Hein, Andriushchenko, and Bitterwolf [51], NNs using this type of activation function produce predictions with arbitrarily high confidence far from the training data. These NNs are representing piecewise affine transformations, dividing the input space into a finite set of convex polytopes. Figure 3.2 shows that the polytopes can extend to infinity, making confidence values produced by NNs unsuitable as a measure of uncertainty. The very same problem even transfers to methods which were targeted at detecting

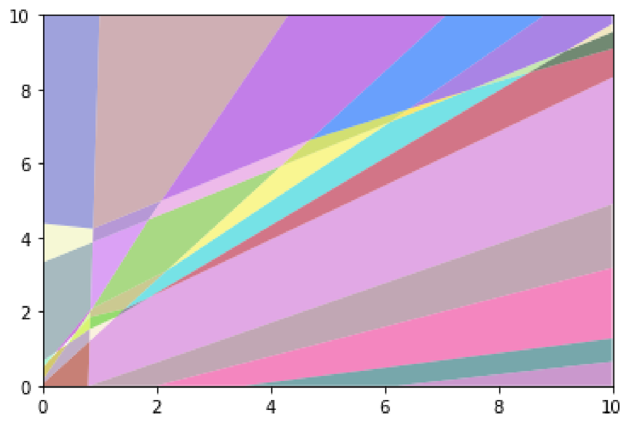

Figure 3.2: Input region within $\mathbb{R}^2$ of a two-hidden layer neural network decomposed into a finite set of polytopes. Graphic is Figure 1 from Hein, Andriushchenko, and Bitterwolf [51].

these OoD inputs, such as temperature rescaling [83] or classifiers with reject option [21, 39]. In the light of this finding and the fact that most modern NNs are affected by this, confidence based decision-making and uncertainty quantification is at high risk. Fortunately, Hein, Andriushchenko, and Bitterwolf [51] mention that there exist models such as Radial Basis Function (RBF) networks, that have the ability to predict a uniform confidence in the OoD regime. They also show that special training techniques, such as enforcing a uniform confidence on adversarial examples, can counteract the overconfidence. Additionally, it might be possible to learn generative models for either the In- or Out-of-Distribution (OoD) data.

3.4 REGULARIZATION

We saw previously that NNs, due to their high capacity, have a universal approximation property. This makes them, however, susceptible to overfitting to the training data and regularization is important in order to achieve generalization when training NNs. Generally speaking, regularization can be seen as restricting the hypothesis space $\mathcal{H}$, or in the context of NNs, favoring specific regions within the weight space. Usually there are no hard restrictions applied to the hypothesis space ('hard' meaning something like "all weights need to lie in the interval of $[-1, 1]$") but rather certain values are given a higher weight than others. This can be described more formally when revisiting the Maximum a Posteriori (MAP) approach. We already viewed the training of NNs from the theory of MLE. In this case we found a set of parameters by maximizing the likelihood of observing the training data within our model

$$\boldsymbol{\theta}^{\text{MLE}} = \arg\max_{\boldsymbol{\theta}} \prod_{(\boldsymbol{x},y)\in\mathcal{S}} \hat{p}(y \mid \boldsymbol{x}, \boldsymbol{\theta})\,. \tag{3.47}$$

MAP however assumes there exists a prior distribution $p(\boldsymbol{\theta})$ over our model parameters and treats $\boldsymbol{\theta}$ as a random variable. We can then find our optimal set of parameters by computing the mode of the posterior distribution over the model parameters

$$\begin{aligned}
\boldsymbol{\theta}^{\text{MAP}} &= \arg\max_{\boldsymbol{\theta}} \prod_{(\boldsymbol{x},y)\in\mathcal{S}} \hat{p}(\boldsymbol{\theta} \mid \boldsymbol{x}, y) && (3.48)\\
&= \arg\max_{\boldsymbol{\theta}} \prod_{(\boldsymbol{x},y)\in\mathcal{S}} \frac{\hat{p}(y \mid \boldsymbol{x}, \boldsymbol{\theta}) p(\boldsymbol{\theta})}{p(y)} && (3.49)\\
&= \arg\max_{\boldsymbol{\theta}} \prod_{(\boldsymbol{x},y)\in\mathcal{S}} \hat{p}(y \mid \boldsymbol{x}, \boldsymbol{\theta}) p(\boldsymbol{\theta}) \,. && (3.50)
\end{aligned}$$

Here, $p(\boldsymbol{\theta})$ expresses our prior knowledge about the model parameters, effectively giving all parameter choices $\boldsymbol{\theta}$ in our hypothesis space $\mathcal{H}$ a weight according to the chosen prior. The denominator $p(y) = \int_{\boldsymbol{\theta}} p(y \mid \boldsymbol{\theta}) p(\boldsymbol{\theta}) \mathrm{d}\boldsymbol{\theta}$ in Eq. (3.49) is called the *marginal likelihood* and can be neglected in the maximization as it does not depend on $\boldsymbol{\theta}$.

When choosing

$$p(\boldsymbol{\theta}) = \mathcal{N}_{d_{\boldsymbol{\theta}}}(\boldsymbol{\theta} \mid 0, \mathbb{I}_{d_{\boldsymbol{\theta}}} \frac{1}{2\lambda}) = \frac{1}{\sqrt{(2\pi \frac{1}{2\lambda})^{d_{\boldsymbol{\theta}}}}} \mathrm{e}^{-\frac{1}{2}\boldsymbol{\theta}^{\mathsf{T}}(\mathbb{I}_{d_{\boldsymbol{\theta}}} \frac{1}{2\lambda})^{-1}\boldsymbol{\theta}} = (\frac{\pi}{\lambda})^{-\frac{d_{\boldsymbol{\theta}}}{2}} \mathrm{e}^{-\lambda \boldsymbol{\theta}^{\mathsf{T}}\boldsymbol{\theta}} \,, \quad (3.51)$$

and a *categorical distribution* on our target data as in Section 3.3, we get

$$\begin{aligned}
\boldsymbol{\theta}^{\text{MAP}} &= \arg\max_{\boldsymbol{\theta}} \prod_{(\boldsymbol{x},y)\in\mathcal{S}} \hat{p}(y \mid \boldsymbol{x}, \boldsymbol{\theta}) p(\boldsymbol{\theta}) && (3.52)\\
&= \arg\max_{\boldsymbol{\theta}} \sum_{(\boldsymbol{x},y)\in\mathcal{S}} \left[\log\left(\hat{p}(y \mid \boldsymbol{x}, \boldsymbol{\theta})\right) + \log\left(p(\boldsymbol{\theta})\right)\right] && (3.53)\\
&= \arg\min_{\boldsymbol{\theta}} \frac{1}{|\mathcal{S}|} \sum_{(\boldsymbol{x},y)\in\mathcal{S}} \left[-\log\left(\hat{p}(y \mid \boldsymbol{x}, \boldsymbol{\theta})\right) + \frac{d_{\boldsymbol{\theta}}}{2} \log\left(\frac{\pi}{\lambda}\right) + \lambda \boldsymbol{\theta}^{\mathsf{T}}\boldsymbol{\theta}\right] && (3.54)\\
&= \underbrace{\arg\min_{\boldsymbol{\theta}} \frac{1}{|\mathcal{S}|} \sum_{(\boldsymbol{x},y)\in\mathcal{S}} -\log\left(\hat{p}(y \mid \boldsymbol{x}, \boldsymbol{\theta})\right)}_{\text{(a)}} + \underbrace{\lambda \left\|\boldsymbol{\theta}\right\|_2^2}_{\text{(b)}} && (3.55)
\end{aligned}$$

Above, (a) is the average *cross-entropy* loss from Eq. (3.23) and (b) is commonly known as *weight decay* [70], with a scaling factor of $\lambda > 0$ and $d_{\boldsymbol{\theta}}$ being the dimensionality of $\boldsymbol{\theta}$. We can see that MAP and ML estimation are very closely related and only differ by the specification of a prior distribution on the model parameters.

As derived above, *weight decay* or also called L_2 regularization, coincides with a (multivariate) Gaussian prior on the weights. Another popular regularization term, as a result of choosing an independent Laplace prior for each weight, is the L_1 regularization

$$\begin{aligned}
p(\boldsymbol{\theta}_i) &= \text{Laplace}(\boldsymbol{\theta}_i \mid 0, \frac{1}{\lambda}) = \frac{\lambda}{2} \mathrm{e}^{-\lambda|\boldsymbol{\theta}_i|} \,, \quad i = 1, \ldots, d_{\boldsymbol{\theta}} \,, && (3.56)\\
p(\boldsymbol{\theta}) &= \prod_{i=1}^{d_{\boldsymbol{\theta}}} \text{Laplace}(\boldsymbol{\theta}_i \mid 0, \frac{1}{\lambda}) = \left(\frac{\lambda}{2}\right)^{d_{\boldsymbol{\theta}}} \mathrm{e}^{-\lambda\|\boldsymbol{\theta}\|_1} \,. && (3.57)
\end{aligned}$$

Which leaves us with the following regularization term in the MAP setting

$$\boldsymbol{\theta}^{\text{MAP}} = \arg\max_{\boldsymbol{\theta}} \prod_{(\boldsymbol{x},y)\in\mathcal{S}} \hat{p}(y \mid \boldsymbol{x},\boldsymbol{\theta})p(\boldsymbol{\theta}) \tag{3.58}$$

$$= \arg\min_{\boldsymbol{\theta}} \frac{1}{|\mathcal{S}|} \sum_{(\boldsymbol{x},y)\in\mathcal{S}} - \log\left(\hat{p}(y \mid \boldsymbol{x},\boldsymbol{\theta})\right) + \lambda\|\boldsymbol{\theta}\|_1 \,. \tag{3.59}$$

In general, L_2 regularization produces small weights while L_1 regularization encourages a sparse weight vector, similar to LASSO regression [130].

Another popular technique for regularization is *dropout* [125]. When using dropout, each neuron output is set to zero with a certain probability during the forward pass, effectively removing it from the computational graph. Intuitively this forces the network to save its knowledge in a redundant way into the weights, so that if one neuron vanishes, the others can compensate for the missing information. From a theoretical perspective, Gal and Ghahramani [37] have shown that training an NN with dropout is a Bayesian approximation of a Gaussian process with a parameterized Bernoulli distribution in place of the intractable posterior. With this we are able to sample from a distribution of weights and thus can also compute a theoretically grounded model uncertainty.

In conclusion, regularization is inevitable in modern DNNs as the large number of parameters carries the risk of overfitting to the training data. On the other hand, restricting the hypothesis space $\mathcal{H}$ might increase the model uncertainty.

3.5 CONVOLUTIONAL NEURAL NETWORKS

3.5.1 *Convolution Operation*

The classic (image) pattern recognition pipeline comprises the preprocessing and feature extraction part before a classification model like an MLP is trained on the extracted features. For a long time these features were computed based on handcrafted heuristics (e.g., SIFT [85], HOG [23] and LBP [101]), which are extracting local image descriptors. The big advantage of modern DNNs does not only come from the universal approximation property but also from the fact that convolutional architectures enable an end-to-end trainable model. They do not rely on handcrafted features because the feature extraction and classification are jointly optimized (see Fig. 3.5 for an illustration). When considering high dimensional data, such as images, a downside of an MLP is its dense architecture. An image of size 256×256 given to an MLP layer with 256^2 input neurons and only 10 output neurons, requires already $655\,370$ weights. Another disadvantage is the sensitivity to scaling and translation of the same input. Every neuron in an MLP is connected to exactly one pixel in the input image, making its receptive field small. Changing the input by translation or scaling should not change the class assignment, but an MLP is affected by these transformations because a single input neuron does not have any contextual information. In contrast to this, the number of weights in a convolutional layer is not dependent on the input's spatial size (except for the number of color channels), greatly reducing the number of parameters. Additionally, the convolution operation makes the sensible assumption that pixels in an image are locally dependent and that only the spatial relation between features (and not the exact position within the image) is relevant to the final prediction. This results in some useful properties of CNNs, such as an invariance to translations and scaling [77]. With the availability of larger datasets and more compute power, end-to-end architectures using convolutional layers greatly improved the performance on image classification benchmarks [69, 77]. The following section will give a short introduction to convolutional layers in DNNs as well as an overview of common architectures.

Mathematically a convolution (depicted by $*$) is an operation on two functions [41, pp. 327–329] $h_1 : \mathbb{R} \to \mathbb{R}$ and $h_2 : \mathbb{R} \to \mathbb{R}$

$$\begin{aligned}(h_1 * h_2)(i) &= \int_{-\infty}^{\infty} h_1(\tau) \cdot h_2(i-\tau)\mathrm{d}\tau \\ &= \int_{-\infty}^{\infty} h_1(i-\tau) \cdot h_2(\tau)\mathrm{d}\tau \\ &= (h_2 * h_1)(i)\,, \quad i \in \mathbb{R}\,.\end{aligned} \tag{3.60}$$

If $h_1 : \mathbb{Z} \to \mathbb{R}$ and $h_2 : \mathbb{Z} \to \mathbb{R}$ are discrete functions the convolution is defined as

$$\begin{aligned}(h_1 * h_2)(i) &= \sum_{k=-\infty}^{\infty} h_1(k) \cdot h_2(i-k) \\ &= \sum_{k=-\infty}^{\infty} h_1(i-k) \cdot h_2(k) \\ &= (h_2 * h_1)(i)\,, \quad i \in \mathbb{Z}\,.\end{aligned} \tag{3.61}$$

Similarly, we can define a convolution for functions on two continuous or discrete variables as

$$\begin{aligned}(h_1 * h_2)(i,j) &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h_1(\tau_1,\tau_2) \cdot h_2(i-\tau_1, j-\tau_2)\mathrm{d}\tau_1\mathrm{d}\tau_2 \\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h_1(i-\tau_1, j-\tau_2) \cdot h_2(\tau_1,\tau_2)\mathrm{d}\tau_1\mathrm{d}\tau_2 \\ &= (h_2 * h_1)(i,j)\,, \quad (i,j) \in \mathbb{R}^2\,,\end{aligned} \tag{3.62}$$

and

$$\begin{aligned}(h_1 * h_2)(i,j) &= \sum_{k=-\infty}^{\infty}\sum_{l=-\infty}^{\infty} h_1(k,l) \cdot h_2(i-k, j-l) \\ &= \sum_{k=-\infty}^{\infty}\sum_{l=-\infty}^{\infty} h_1(i-k, j-l) \cdot h_2(k,l) \\ &= (h_2 * h_1)(i,j)\,, \quad (i,j) \in \mathbb{Z}^2\,.\end{aligned} \tag{3.63}$$

In the context of image data we have a two or three-dimensional input, depending on whether we have a colored (RGB) or gray scale image. For example, for gray

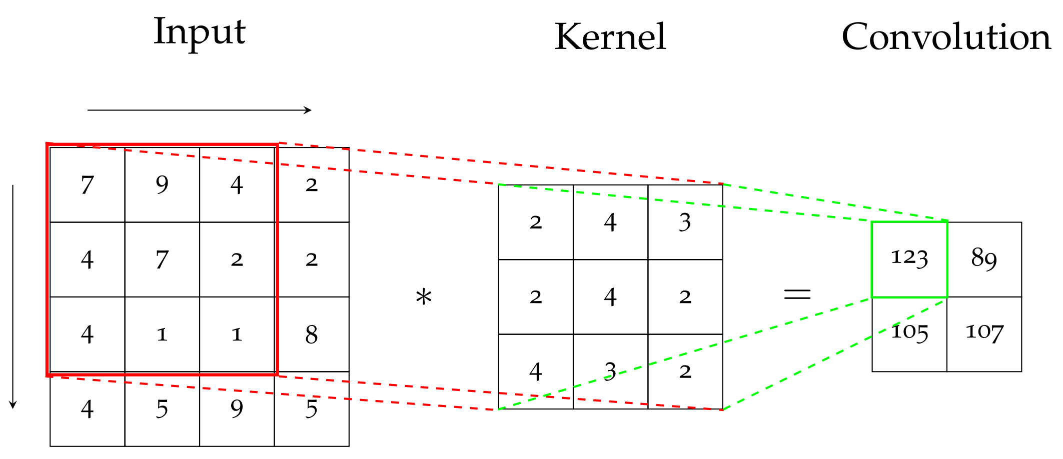

Figure 3.3: Convolution operation on a 4×4 input with a 3×3 kernel and a stride of 1.

scale images, Eq. (3.63) is applicable as we can represent the input image as a function $h_1 : \mathbb{N}^2 \to \{0, 1, \ldots, 255\}$, mapping pixel coordinates (i, j) to gray scale values $\{0, 1, \ldots, 255\}$. Intuitively speaking, a convolution slides a kernel with a predefined size over the image, computing a weighted sum between the kernel weights and the pixel values (see Fig. 3.3 for an illustration). The step size of the convolution is also called *stride* and is a hyperparameter which has to be defined in advance. Because the same kernel weights are applied at each valid position of the input image, the weights are effectively shared, thus drastically reducing the number of parameters. The convolution also makes the feature extraction invariant to translation as the same filter is applied at each position of the image.

3.5.2 *Pooling*

Pooling is another important building block in modern convolutional architectures. The operation aggregates input values within a predefined window, and similar to the case of a convolutional layer, this window slides over the input. The classical aggregation function is the maximum which only considers the neuron with the highest activation within the current window (see Fig. 3.4 for an illustration). Pooling reduces the dimensionality of the input, lowering the number of computing operations, and increases the size of the *receptive field* for subsequent neurons. The *receptive field* are all input values that take part in computing the activation of a neuron. As the pooling and convolution operation reduce the spatial size of the input, neurons that are closer to the output of the network are influenced by a larger part of the input, resulting in a larger *receptive field*. Besides increasing the receptive field, pooling also makes the features robust in terms of translation. The exact position of a feature inside the image does not matter, only the relative position to other features is of interest for the final classification. As mentioned above, fully convolutional architectures for image classification tasks often make use of a special pooling type called *Spatial Pyramid Pooling* (SPP) [49]. SPP uses windows of variable

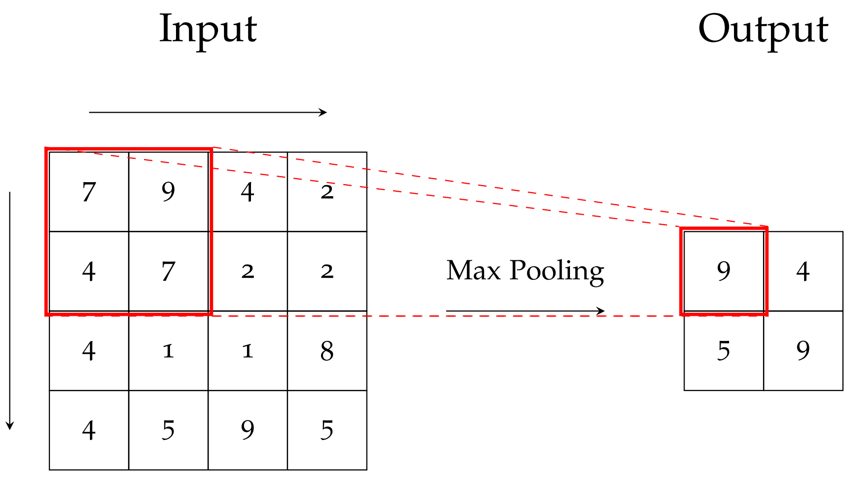

Figure 3.4: Max-Pooling with a 2 × 2 window size and a stride of 2.

size to be able to place a grid with a fixed number of cells over the input. Grids of variable size will then be applied simultaneously to the input and the pooled output will be concatenated. This allows the SPP layer to have an output of a fixed size, enabling the NN to process inputs of arbitrary size.

3.6 COMMON ARCHITECTURES

In the last decades a number of NN architectures made significant progress on different tasks and application domains. Some components in these models have become standard building blocks of modern NNs. Among them are, e.g., the LeNet [77], AlexNet [69], VGG [122], ResNet [50], DenseNet [59] and Transformer [134]. This list is by no means complete, but it describes a cross-section of DNN research over the last few decades. All aforementioned architectures proposed changes which have significantly advanced the state-of-the-art in NN pattern recognition. The LeNet architecture by Lecun et al. [77] was one of the first convolutional architectures that made a large improvement on the task of object classification. Although it is very small with a total of 60 000 trainable parameters, the network showed a remarkable classification accuracy in comparison to other common methods such as nearest neighbors, polynomial classifiers, RBF networks, MLPs or Support Vector Machines (SVMs). With the availability of the ImageNet dataset [27] and the utilization of Graphics Processing Unit (GPU) accelerated training of DNNs, AlexNet [69] received a lot of attention when it won the ILSVRC-2012 image classification challenge with a relative improvement of 42 % over the second-best competitor. This large improvement was achieved by a very deep CNN, which could only be trained by splitting it in multiple parts in order to be able to train on multiple GPUs in parallel. Simonyan and Zisserman [122] explored even deeper convolutional architectures and won the ILSVRC-2014 image classification challenge with their VGG model. Compared to LeNet (60 K) and AlexNet (60 M), the VGG-19 model has

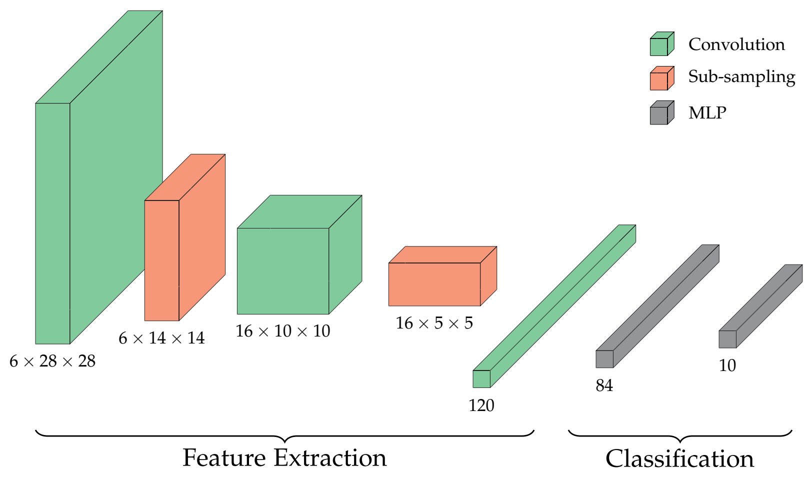

Figure 3.5: LeNet architecture by Lecun et al. [77]. The feature map sizes after each operation are depicted under the respective blocks. Note that the activation functions are omitted and should be applied after each layer. The network was designed to be trained on 32 × 32 sized input images.

144 M parameters. Contrary to the development of NNs with ever more weights, He et al. [50] proposed Residual Neural Networks (ResNets) and won the ILSVRC-2015 image classification challenge with only 1.7 M parameters. The results were achieved by adding additional connections to subsequent layers, skipping multiple of them in between (more on that in Section 3.6.2). Inspired by the findings of He et al. [50], Huang et al. [59] proposed dense convolutional networks (DenseNets) that take the idea of residual connections to the extreme, connecting each layer to all previous layers. With DenseNet they were able to reduce the amount of weights, while significantly decreasing the error rates on several image classification benchmarks compared to ResNets. Recently, autoregressive language models such as GPT-3 [13] or LaMDA [129] with up to 175 B parameters, have caught attention. They can be trained unsupervised on large text corpora and fuel other methods such as text-to-image synthesizers [104, 111]. Modern language models have in common that they utilize the transformer architecture [134], which is specialized on sequential data such as text. Although transformers were originally proposed for sequential models, they have been successfully applied to vision problems, like in the case of Vision Transformers (ViT) [31]. In the following sections, we will explore some of the aforementioned architectures in more detail, as we will utilize them later in the experiments.

3.6.1 *LeNet*

In the work by Lecun et al. [77], the network was evaluated on the MNIST dataset of handwritten digits, which was published in the very same work and is still used today for small experiments and proofs of concept. Figure 3.5 illustrates the architecture. Built upon the most basic building blocks, the network consists

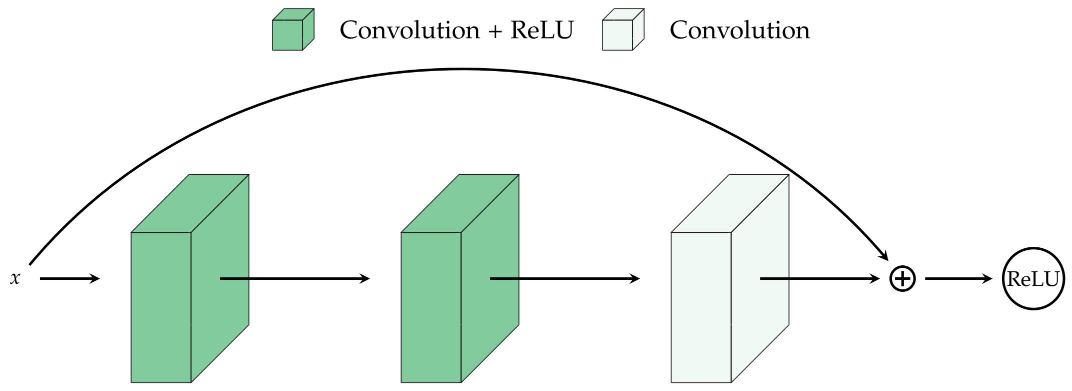

Figure 3.6: Residual Neural Network (ResNet) block as in He et al. [50] with a total of 3 convolutional layers.

of three convolutional layers with kernel sizes of $6 \times 5 \times 5$, $6 \times 16 \times 5 \times 5$ and $16 \times 120 \times 5 \times 5$, respectively. The first two convolutional layers are followed by a sub-sampling layer with a window size of 2×2 and a stride of 2, hence the individual receptive fields are non-overlapping. According to Lecun et al. [77], "the four inputs to a unit [...] are added, then multiplied by a trainable coefficient, and then added to a trainable bias. The result is passed through a sigmoidal function" [77, pp. 7–8]. When used on input images of size 32×32, the feature map size after the last convolution is $120 \times 1 \times 1$, making it possible to directly feed the feature map into an MLP. The final classification part has a hidden layer with 84 neurons and 10 output neurons for the 10 digit classes in the MNIST dataset. As activation function the authors use a scaled hyperbolic tangent

$$\phi(a) = 1.7159 \cdot \tanh(\frac{2a}{3}) \tag{3.64}$$

3.6.2 *Residual Networks*

As briefly mentioned above, ResNets achieve superior performance compared to other DNN architectures, while reducing the number of parameters. This is achieved by learning residual functions inside the network. Let $\mathfrak{B}$ represent an arbitrary number of subsequent layers inside the network (also called *block*) with corresponding weights $\boldsymbol{\theta}_{\mathfrak{B}}$. The residual learning function can then be described as

$$\boldsymbol{o} = \mathrm{ReLU}(\mathfrak{B}(\boldsymbol{x} \mid \boldsymbol{\theta}_{\mathfrak{B}}) + \boldsymbol{x})\,, \tag{3.65}$$

with $\boldsymbol{x}$ and $\boldsymbol{o}$ the input and output of the block, respectively. An illustration of this can be seen in Fig. 3.6. Note that in order to be able to compute the element-wise summation, $\mathfrak{B}(\boldsymbol{x} \mid \boldsymbol{\theta}_{\mathfrak{B}})$ and $\boldsymbol{x}$ need to have the same dimension. If this is not the case, $\boldsymbol{x}$ can be linearly projected into the same space by learning a corresponding projection matrix. The reasoning behind this residual formulation is the degradation problem described by He et al. [50]. In their work, the authors showed that deeper

Layer Name	Output Size	ResNet 18	ResNet 101
conv1	112×112	7×7, 64, stride 2	
	56×56	3×3 max pool, stride 2	
conv2_x	56×56	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
conv3_x	28×28	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$
conv4_x	14×14	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 23$
conv5_x	7×7	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
	1×1	average pool, 1000-d fc, softmax	
FLOPs		1.8×10^9	7.6×10^9

Table 3.1: Number of residual blocks of ResNet 18 and ResNet 101 architectures. This table is a part of table 1 in He et al. [50, p. 5].

network architectures which were built sequentially, had a higher training and testing error than their shallower counterparts. This is counterintuitive at first, since more parameters should correspond to higher capacity and hence smaller training error. In theory, the training error of deep networks should not be higher than for the shallow networks as the extra layers could be added as identity mapping, restoring the original model. The authors argue that this effect is unlikely to be caused by a vanishing gradient, as they observed a typical size for the norm of the gradient. As a conclusion they argue that the degradation is caused by other unspecified optimization problems.

A complete ResNet architecture is formed by stacking multiple of these residual blocks on top of each other. Table 3.1 shows two network configurations for the ResNet 18 and ResNet 101. The former is built with blocks containing two convolutional layers, while the latter has an additional convolution at the end of each block. Also, the number of filters in each block differ significantly, which is also apparent in the number of Floating Point Operations (FLOPs) needed for each model.

4 RELATED WORK

This chapter gives an overview of related works in the field of *Uncertainty Quantification* (UQ), adversarial examples as well as tasks which rely on uncertainty estimates. Abdar et al. [1] and Gawlikowski et al. [38] mention the areas of Bayesian and ensemble learning, while Gawlikowski et al. [38] additionally mention "Single Network Deterministic Methods". The following sections will give a compressed overview of these fields while additionally including a section covering generative methods, which emerged in recent years. Therefore, methodologies in the area of UQ can be grouped into four main categories: Frequentist, Bayesian, Ensemble and Generative approaches. We will additionally cover the topic of *Adversarial Examples*, which is not directly related to the field of UQ, but which poses a threat on DNNs during inference.

4.1 FREQUENTIST

Frequentist Approaches only need a single, or at most two, forward passes to compute uncertainty estimates, which makes them especially interesting in resource constraint applications. These methods can be further distinguished into *external* and *internal*, which will be discussed in the following sections.

4.1.1 *External Methods*

External methods do not have an influence on the network architecture or training process and can often be applied to existing models [20, 47, 52, 80, 83]. A widely adopted baseline from the area of confidence estimation is the Maximum Class Probability (MCP), proposed by Hendrycks and Gimpel [52]. As the name implies, MCP takes the class probability of the predicted class as a measure of confidence. MCP is easy to interpret and compute, however, Hendrycks and Gimpel [52] also show that modern neural networks are often overconfident in their predictions, assigning a high MCP to unknown or wrongly classified examples. They show in their experiments that an image classifier trained on the MNIST dataset [77] had an average MCP of 91 % on randomly sampled Gaussian noise. This demonstrates that this confidence measure is in general incapable of detecting unknown data points. The same holds true for misclassified examples, where an average MCP of 86 % could be observed. Besides the implications of piecewise linear activation functions (see Section 3.3.5), the softmax activation function (Eq. (3.9)), which uses an exponential scaling to form a smooth approximation of an indicator function, also causes overconfidence. For this activation function, small changes to the input can lead to large shifts in the resulting class distribution.

$$\begin{aligned} \boldsymbol{f}(\boldsymbol{x} \mid \boldsymbol{\theta})_1 &= 97 \;\; \Rightarrow \hat{p}(y=1 \mid \boldsymbol{x}) = \frac{e^{97}}{e^{97}+e^{100}} \approx 4.74\,\% \\ \boldsymbol{f}(\boldsymbol{x} \mid \boldsymbol{\theta})_2 &= 100 \Rightarrow \hat{p}(y=2 \mid \boldsymbol{x}) = \frac{e^{100}}{e^{97}+e^{100}} \approx 95.26\,\% \end{aligned} \tag{4.1}$$

Equation (4.1) is demonstrating this in a simple scenario with two classes. We can see that although the difference in the logits is small, applying the softmax function on them results in more than 95 % confidence on class 2. In the above example both logits are relatively large, which could be the result of examples with high data uncertainty and features which match both classes. However, even in the case with logits of $f_1(\boldsymbol{x} \mid \boldsymbol{\theta}) = 2$ and $f_2(\boldsymbol{x} \mid \boldsymbol{\theta}) = 0$, which is essentially an input that does not match to any of the seen features, the first class is assigned a confidence of over 88 %. As a human this seems unreasonable, and we would expect the confidence to be lower in both cases. Following works try to mitigate this problem by applying different calibration techniques on the model output [47]. A popular approach is *temperature scaling*, where the last layer outputs are scaled by a factor $\frac{1}{T}$ before the softmax activation function is applied

$$\text{(temperature scaling)} \quad \frac{e^{\boldsymbol{f}(\boldsymbol{x}|\boldsymbol{\theta})_i/T}}{\sum_{j=1}^{K} e^{\boldsymbol{f}(\boldsymbol{x}|\boldsymbol{\theta})_j/T}}, \quad \forall i = 1, \ldots, K. \tag{4.2}$$

Here, T is called the *temperature coefficient*. For $T > 1$ the output of the softmax is smoothed out, which increases the overall entropy and reduces overconfident predictions. The value of T is determined empirically on a validation set by minimizing the Negative Log-Likelihood Loss (NLLL). It is important to note that this technique reduces the calibration error (see Chapter 7) resulting from overconfident predictions but does not improve the ability to separate In- from Out-of-Distribution (or correctly from wrongly predicted) examples. Liang, Li, and Srikant [83] utilize the same approach but additionally apply an adversarial preprocessing to the inputs to increase their softmax score. Their experiments reveal that this preprocessing has a greater impact on ID than on OoD inputs, improving the separability between the two. They term this approach ODIN. The authors of [80] fit class-conditional Gaussian distributions on the output of the penultimate layer. Their confidence score is then derived by computing the Mahalanobis distance to the nearest class-conditional Gaussian. They argue that computing a confidence score on the penultimate layer alleviates the problem of overconfident predictions caused by the softmax activation function.

An external model to estimate the prediction confidence is used in Corbière et al. [20]. Their auxiliary network predicts the True Class Probability (TCP) of the classification model. The TCP is the softmax output corresponding to the annotated class. For correct predictions this is equivalent to the MCP but differs in the case of wrong classifications. Due to the fact that one does not know the true class during inference, they are training their auxiliary model on the true class probabilities gathered from the training or validation dataset. They evaluate their confidence score on the task of FP detection and domain adaptation based on self-training. The problem of OoD detection is not addressed as the TCP is not defined for unknown object classes.

4.1.2 *Internal Methods*

Internal approaches are altering the network structure or training process of the classification model. Depending on the architecture or application, some of these methods might be incompatible.

A common approach is to extend the classification model with an additional class for the OoD domain, which is also called *classifier with reject option*. This can either be done by explicitly adding this class to the output of the model [21, 39] or implicitly by optimizing a different loss for unknown examples on the already known classes [51, 53]. For example, Hendrycks, Mazeika, and Dietterich [53] minimize the cross entropy to a uniform distribution, which is equivalent to maximizing the entropy on the OoD objects. This enables the model to reject the classification of an example by assigning a higher entropy to it if it does not belong to the known classes. Some works argue that using a confidence loss instead of a reject class is more suitable [51, 53, 79] while others are arguing that using an explicit reject class is increasing the generalization to unknown regions in the input space [136]. A problem that arises in both of these settings is the choice of training data for the OoD domain. Hendrycks, Mazeika, and Dietterich [53] use real world auxiliary datasets as a proxy for the OoD domain while Hein, Andriushchenko, and Bitterwolf [51] use augmented ID examples and Gaussian noise. Utilizing adversarial examples as a placeholder for unknown inputs is also a common strategy [75, 80]. In practically all learning settings, the OoD domain can be considered as an open set and thus being infinitely diverse. Aside from the fact that we need to collect or generate additional data, we will also never be able to cover the whole OoD domain, which is a shortcoming of these approaches.

Closely related to the principle of a reject class is the work by DeVries and Taylor [29]. They augment the model architecture by adding a second head which predicts a confidence score. The confidence score is learned by allowing the network to interpolate between its own prediction and the given annotation during the training phase. The predicted confidence serves as the interpolation factor. During test time the output of the second branch is used as the prediction confidence. Two output heads are also used by Hsu et al. [57], where the authors propose a decomposed confidence based on the rule of conditional probabilities. Their approach can be seen as a generalization to ODIN [83], where the temperature scaling parameter (see Eq. (4.2)) is learned by the second head of the network. Ren et al. [105] also approach the problem of confidence estimation by separating it into a ratio of semantic and general background statistic. For estimating the two components they employ two models. The first model is trained on the original ID data, while the second one is receiving perturbed examples. The intuition with this approach is that if the examples are sufficiently noisy, they lose the semantic information and the model only learns the background statistics. During test time the logarithmic ratio between these two probabilities serves as a confidence score. Although most of the empirical results indicate an improvement over the commonly used baseline by Hendrycks and Gimpel [52], the experimental settings – in terms of dataset selection – remain limited in these works.

An OvA classifier is an approach which alters the training procedure of an NN but not the architecture. In practice, OvA classifiers can be realized by an ensemble of independent binary classifiers, or by applying the sigmoid instead of softmax activation function to the output of a single neural network. The latter case has the same network architecture as a softmax based NN but resembles K OvA classifiers which are sharing the feature representations. Note that even in the first case, where the OvA classifier is realized as an ensemble of K independent binary NNs, these techniques do in general not belong to the class of ensemble uncertainty quantification methods. Uncertainty quantification using ensembles requires the individual ensemble members to solve the same task, which is not true for the OvA model. Using OvA classifiers gives the advantage that a part of the training dataset can be used for the "all" part, which then corresponds to an OoC score. A second benefit is that the individual binary classifiers are not forced into a closed world with a fixed label set, which has the potential to improve the over-confidence issue with all-vs-all models. Padhy et al. [102] explore this possibility in the context of distance-based logits, where instead of applying an affine transformation followed by the softmax function on the last layer, the network is learning a class prototype and encoding the logits as the distance to it. The mapping in the $[0, 1]$ domain is then performed by applying class-wise sigmoid activations. In experiments this results in a more well-behaved prediction confidence and alleviates the pathological overconfidence of softmax based all-vs-all classifiers. The loss formulation in this setup could however lead to an imbalance in the case of many classes, as for each binary classifier the loss corresponding to "all" is summed over all remaining classes. A question that arises in *One-versus-All* (OvA) settings is about the way the individual binary classifiers are aggregated to form a joint prediction. Padhy et al. [102] use the *winner-takes-it-all* approach and choose the binary classifier with the highest confidence. However, in the work by Franchi et al. [35], the aggregation is performed by learning a weight for each OvA classifier. This is done by training an all-vs-all classifier as well as K binary classifiers jointly, where the confidence of the all-vs-all classifier is corresponding to the weight of the respective binary classifier in the final model. The problem with the imbalanced OvA loss in Padhy et al. [102] can also be circumvented by doing a hard negative mining on the negative cases. This is exactly what Saito and Saenko [112] are proposing in their work. They term it *Hard Negative Classifier Sampling* and use the resulting OvA classifiers to detect unknown samples in a domain adaptation setting. In a setup where the target domain contains a number of unknown labels they improve the detection of these unknown samples and thus increase the performance on the known targets.

The theory of evidence and subjective logic is another way of reformulating the training of NNs without using a softmax activation function and was explored under the topics of evidential deep learning [118] and posterior networks [18]. These works describe the prediction of an NN as a sample of a Dirichlet distribution. Instead of directly predicting class labels, the model estimates the evidence for a particular class, which then translates to the concentration parameters of a Dirichlet distribution. The Dirichlet distribution is the natural (conjugate) prior for a categorical distribution and allows to express both aleatoric and epistemic uncertainty with a single forward pass. Given the concentration parameters $\boldsymbol{\alpha}_1, \dots, \boldsymbol{\alpha}_K$ for an input $\boldsymbol{x}$, the probability of $\boldsymbol{x}$ being of class y can be computed as in [118]

$$\hat{p}(y = k \mid \boldsymbol{x}, \boldsymbol{\alpha}) = \frac{\boldsymbol{\alpha}_k}{\mathcal{E}}, \tag{4.3}$$

$$\mathcal{E} = \sum_{k=1}^{K} \boldsymbol{\alpha}_k \,. \tag{4.4}$$

Following this, the aleatoric and epistemic uncertainties for the class prediction of $\boldsymbol{x}$ is given by

$$\text{(aleatoric)} \quad u_a = H(\hat{p}(y \mid \boldsymbol{x}, \boldsymbol{\alpha})) \tag{4.5}$$

$$= -\sum_{k=1}^{K} \hat{p}(y = k \mid \boldsymbol{x}, \boldsymbol{\alpha}) \cdot \log(\hat{p}(y = k \mid \boldsymbol{x}, \boldsymbol{\alpha}))\,, \tag{4.6}$$

$$\text{(epistemic)} \quad u_e = \frac{K}{\mathcal{E}} \,. \tag{4.7}$$

Sensoy, Kaplan, and Kandemir [118] replace the softmax activation function with a strictly positive one which enables the interpretation of the output as the evidence for the respective classes. The concentration parameters are then computed as

$$\boldsymbol{\alpha}_k = \boldsymbol{f}(\boldsymbol{x} \mid \boldsymbol{\theta})_k + 1, \quad \forall k = 1, \dots, K \,. \tag{4.8}$$

They train their NN with an L_2 loss between class label and $\hat{p}$ from Eq. (4.3) and additionally add a regularization loss in form of the KLD between a uniform Dirichlet ($\boldsymbol{\alpha}_k = 1, \forall k = 1, \dots, K$) and the predicted one. Instead of directly predicting the evidence with a neural network, Charpentier, Zügner, and Günnemann [18] train an encoder network and a class conditional normalizing flow within the resulting latent space. The normalizing flow predicts probability densities for each class, which are multiplied by the number of examples in the training set to get the evidence for the respective class. Similar to Sensoy, Kaplan, and Kandemir [118], they apply a regularizer based on the KLD between the predicted and a uniform Dirichlet distribution.

4.2 BAYESIAN

Recalling Section 2.2, Bayesian approaches have a probabilistic view on the world and place a distribution on model parameters

$$p(\boldsymbol{\theta} \mid \mathcal{D}) = \frac{p(\mathcal{D} \mid \boldsymbol{\theta}) p(\boldsymbol{\theta})}{p(\mathcal{D})} . \tag{2.14}$$

The predictive posterior is computed by integrating over all model weights

$$p(y \mid \boldsymbol{x}, \mathcal{D}) = \int p(y \mid \boldsymbol{x}, \boldsymbol{\theta}) p(\boldsymbol{\theta} \mid \mathcal{D}) \mathrm{d}\boldsymbol{\theta} . \tag{2.15}$$

In practical applications this formulation has multiple problems. First, the computation of this integral is intractable and needs to be approximated with, e.g., MC sampling. Second, the true posterior $p(\boldsymbol{\theta} \mid \mathcal{D})$ is usually too complex to be learned exactly. This is why in most of the BNN literature, these two problems are addressed [9, 36, 37, 138]. Welling and Teh [138] use a stochastic gradient Markow-Chain-Monte-Carlo (SG-MCMC) to approximate samples from the true posterior distribution $p(\boldsymbol{\theta} \mid \mathcal{D})$. This work solved a problem of Markow-Chain-Monte-Carlo (MCMC) methods, which had to calculate the gradient over the entire dataset, making training complex BNNs in large datasets computationally expensive. Another way of approximating the potentially very complex posterior distribution is by utilizing variational inference methods. *Bayes-by-Backprop* is such a method proposed by Blundell et al. [9]. In variational inference the intractable posterior distribution $p(\boldsymbol{\theta} \mid \mathcal{D})$ is replaced by a family of parameterized distributions $q(\boldsymbol{\theta} \mid \boldsymbol{\vartheta})$. This allows to optimize over the parameters of the variational distribution and reduces the complexity. The training objective is then constructed by computing the KLD between the true and variational posterior

$$\begin{aligned} \boldsymbol{\vartheta}^* &= \arg\min_{\boldsymbol{\vartheta}} \mathrm{KL}(q(\boldsymbol{\theta} \mid \boldsymbol{\vartheta}) \parallel p(\boldsymbol{\theta} \mid \mathcal{D})) && (4.9) \\ &= \arg\min_{\boldsymbol{\vartheta}} \int q(\boldsymbol{\theta} \mid \boldsymbol{\vartheta}) \log \frac{q(\boldsymbol{\theta} \mid \boldsymbol{\vartheta})}{p(\boldsymbol{\theta}) p(\mathcal{D} \mid \boldsymbol{\theta})} \mathrm{d}\boldsymbol{\theta} && (4.10) \\ &= \arg\min_{\boldsymbol{\vartheta}} \underbrace{\mathrm{KL}(q(\boldsymbol{\theta} \mid \boldsymbol{\vartheta}) \parallel p(\boldsymbol{\theta}))}_{\text{(a)}} - \underbrace{\mathbb{E}_{q(\boldsymbol{\theta} \mid \boldsymbol{\vartheta})} \left[\log(p(\mathcal{D} \mid \boldsymbol{\theta})) \right]}_{\text{(b)}} . && (4.11) \end{aligned}$$

Equation (4.11) is called the variational or Evidence Lower Bound (ELBO) [65], which is composed of a data-dependent likelihood cost (b) and the complexity cost (a) (cf. [9, p. 3]). The likelihood cost is responsible for fitting the distribution of NN weights to our training data, while the complexity cost functions as a regularizer, keeping the variational posterior close to a given prior $p(\boldsymbol{\theta})$. In most cases the variational posterior is chosen from the *exponential family*, which has nice properties and often allows computing closed form solutions of Eq. (4.11) (a) when choosing a conjugate prior. When choosing a diagonal Gaussian distribution as the variational posterior, training a BNN can be accomplished by first sampling a set of weights

$$\begin{aligned} \boldsymbol{\vartheta} &= (\boldsymbol{\mu}, \rho) , && (4.12) \\ \epsilon &\sim \mathcal{N}(0, I) , && (4.13) \\ \boldsymbol{\theta} &= \boldsymbol{\mu} + \log(1 + \exp(\rho)) \cdot \epsilon . && (4.14) \end{aligned}$$

Afterwards the ELBO can be approximated by using a number of Monte Carlo (MC) samples. For a single sample as above, the ELBO can be computed as

$$\mathfrak{L}(\boldsymbol{\theta}, \boldsymbol{\vartheta}, \mathcal{D}) = \log(q(\boldsymbol{\theta} \mid \boldsymbol{\vartheta})) - \log(p(\boldsymbol{\theta})) - \log(p(\mathcal{D} \mid \boldsymbol{\theta})) , \tag{4.15}$$

where $-\log(p(\mathcal{D} \mid \boldsymbol{\theta}))$ breaks down to the cross-entropy loss as described in Section 3.3.1. When the prior is also chosen to be a diagonal Gaussian, then the complexity cost can be computed in a closed form without requiring MC sampling. Finally, we can do gradient descent on the parameters of the variational distribution instead of the weights themselves

$$\begin{aligned} \boldsymbol{\mu} &\leftarrow \boldsymbol{\mu} - \eta \frac{\partial \mathfrak{L}(\boldsymbol{\theta}, \boldsymbol{\vartheta}, \mathcal{D})}{\partial \boldsymbol{\mu}} , \\ \rho &\leftarrow \rho - \eta \frac{\partial \mathfrak{L}(\boldsymbol{\theta}, \boldsymbol{\vartheta}, \mathcal{D})}{\partial \rho} . \end{aligned} \tag{4.16}$$

Equations (4.12) to (4.14) are called a reparameterization trick [65, p. 4] and allows the direct backpropagation to the distributional parameters.

Gal and Ghahramani [37] show that the widely used regularization technique *dropout* [125] can be seen as an approximation of a Gaussian process. By applying dropout before each weight layer – therefore using a Bernoulli distribution as the variational distribution – we can approximate the weight posterior distribution. Using dropout we do not have to make large architectural changes and still get the benefits of Bayesian inference.

All these methods gather a set of T parameters $\{\boldsymbol{\theta}^t\}_{t=1}^T$ from the posterior distribution and compute the empirical predictive posterior via Monte-Carlo integration

$$p(y \mid \boldsymbol{x}, \mathcal{D}) \approx \frac{1}{T} \sum_{t=1}^{T} p(y \mid \boldsymbol{x}, \boldsymbol{\theta}^t) . \tag{4.17}$$

The same applies for the computation of the aleatoric and epistemic uncertainty as proposed by Depeweg et al. [28] (see Section 2.2, Eqs. (2.17) to (2.19)). This very same methodology can also be utilized to separate different uncertainties in the case of ensembles, which will be discussed next.

4.3 ENSEMBLE

Ensemble Approaches make use of the highly non-linear loss landscapes of DNNs, which have a diverse set of local minima. Random effects such as weight initialization, training data augmentation or batch sampling allows the model to converge to different local extrema. Uncertainty is then computed by calculating the variance in the individual predictions or the entropy over the ensemble, as described in Section 2.2. Many ensemble schemes exist, such as bagging, boosting, or knowledge

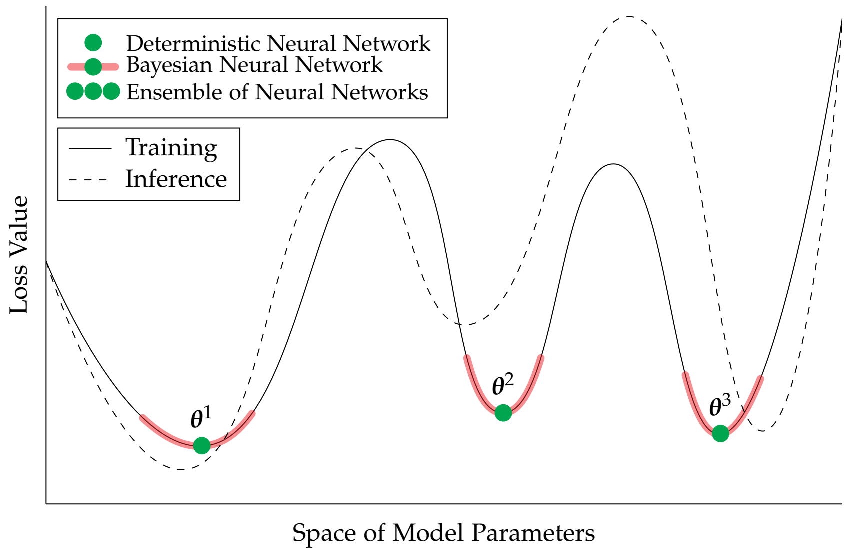

Figure 4.1: Visualization of the learned parameters by deterministic (frequentist), Bayesian and ensembles of neural networks. This figure is based on Figure 7 from Gawlikowski et al. [38, p. 18].

distillation. Note that in order to compute uncertainties using ensembles, each ensemble member needs to solve the same task. The respective ensemble members can receive different training datasets but need to predict the same label set. This is also why OvA classifiers, which can be realized as an ensemble of binary classifiers, are not considered to be part of this category. In practice, these methods are straight forward to apply, require no change in model architecture, and are often easy to parallelize. However, the disadvantages are similar to Bayesian approaches, where training and evaluating ensembles requires significantly more computation than for a single network. Ensembles also consume a larger amount of memory, as multiple network configurations need to be kept available during inference, making them less suitable for resource constrained applications. Although ensembling is conceptually similar to Bayesian model averaging, the resulting model is quite different. As pointed out by Lakshminarayanan, Pritzel, and Blundell [75], even when we can perform exact Bayesian inference, the final model can still suffer from mis-specification. Illustrated in Fig. 4.1, Bayesian neural networks perform model averaging around a local mode in the loss landscape, while ensembles are able to be more diverse because of the combination of different local minima. This makes them in turn more robust to domain-shifts between the training and inference phases. The theories for improving weak learners with ensemble techniques such as, e.g., *boosting* [32, 115] or *bagging* [11] have been around for a long time. One of the first works on predictive uncertainty quantification using ensembles of neural networks was done by Lakshminarayanan, Pritzel, and Blundell [75]. Their simple approach of averaging over multiple independently trained neural networks showed improved performance in the overall accuracy and in detecting unknown classes. The large amount of computational resources needed to train DNNs can be a problem when trying to use ensemble techniques. This is why Huang et al. [58] use a cyclic learning rate schedule to find multiple local minima during a single training run. By rapidly increasing and then decaying the learning rate using a cosine function, the SGD algorithm can escape from local minima. The ensemble is then built over previously visited minima, which is why they call this approach *Snapshot Ensembling*. By using this method, the training time of the ensemble is greatly reduced while still achieving the highest possible benefits. Izmailov et al. [63] use a similar approach called Stochastic Weight Averaging (SWA) and average weight configurations along the SGD optimization trajectory, which improves the generalization without any additional computational overhead during test time. Maddox et al. [87] connect ensemble techniques and Bayesian inference by estimating second order moments during SWA, in order to approximate a Gaussian Distribution over weights (SWA-Gaussian (SWAG)). Empirically the learned Gaussian is approximating the real posterior distribution reasonably well. During test time, several weight configurations are sampled from the Gaussian and averaged. The prediction uncertainty can then be obtained from the output variability as previously described.

4.4 GENERATIVE

In recent years generative methods such as, e.g., autoencoders [54, 65] and GANs [12, 146], have shown an astonishing performance on the representation learning and data synthesis tasks. As these models are approximating the data distribution and its density, they also became more interesting to the UQ community [67, 79, 116, 117, 124, 126, 136, 139] and set new state-of-the art in OoD and FP detection. Xia et al. [139] use the reconstruction error of an autoencoder to detect outliers (OoD objects) and they show empirically that the reconstruction error for OoD objects is considerably higher than for ID ones. This makes the reconstruction error a suitable confidence measure for the task of OoD detection. Schlegl et al. [116] are using a GAN to detect anomalies during inference, which can be seen as a form of OoD detection. First, the GAN is trained on clean data, which results in the generator learning a mapping from a latent space to the image space given by the training data. During inference, they try to find an embedding in the latent space of the generator, for which the generated image is visually close to the given one. As generators are usually not invertible the latent embedding is optimized by performing gradient descent on a reconstruction loss between the generated and real image. If the given image is an outlier and thus not following the training data distribution, the generator is unlikely to be able to generate a similar image, resulting in a large residual loss. This loss can then be used as a confidence measure for detecting OoD examples. Kong and Ramanan [67] also utilize a GAN architecture for OoD detection. Instead of using the generator, they use the resulting discriminator, which is already trained to discriminate between training samples and artificial ones. Because of the instable training of GANs they need to continuously validate the discriminator's OoD detection performance on a holdout OoD set in order to select a checkpoint that performs well in the OoD detection task. Another direction using

GANs was explored by Lee et al. [79], where a GAN was trained to generate data for the OoD domain. The classification model is then trained to maximize the prediction entropy on these generated samples. By integrating the classification model into the GAN training and playing a min-max-min game between generator, discriminator and classifier, the authors aim to generate samples on the boundary of the training distribution. Sricharan and Srivastava [124] build upon the findings of Lee et al. [79] and improve the OoD detection results by slightly tweaking the GAN and classifier training objective. However, further analysis of Vernekar et al. [136] showed that the entropy maximization is not sufficient to generalize to distant regions in the OoD domain and that the GAN setup by Lee et al. [79] suffers from mode collapse. They in turn propose in [135] to use a Variational Autoencoder (VAE) in order to learn a low-dimensional Gaussian distribution which approximates the training data. The VAE can then be used to generate two types of OoD data. Type I samples are generated by applying a perturbation which is perpendicular to the tangent of the VAE latent space. They are similar to adversarial examples and can be used to harden the classification model against these. Type II samples are generated on the contour line of the Gaussian learned by the VAE, which encompasses 95 % of the training data. The contour line is defined by searching for the Mahalanobis distance to the learned Gaussian distribution, satisfying the criteria of containing 95 % of the training data. The generated samples are then used to train a reject classifier, from which the softmax output of the reject class is used as a confidence measure for the task of OoD detection. A VAE is also utilized by Sensoy et al. [117], where the authors train a GAN inside the latent space of the VAE. An advantage of training in the latent space is the dimensionality reduction, which makes it easier to approximate the boundary of the training distribution. Sensoy et al. [117] use the mean predicted by the VAE and let a generator output a variance so that the points sampled from the resulting Gaussian are close to the training data in the latent space but still distinguishable from the real data in the image space. They achieve this by using a similar GAN training setup as Lee et al. [79].

4.5 ADVERSARIAL EXAMPLES

Adversarial examples are perturbed examples such that a classifier assigns the perturbed example to a different class than the original one. They can be further categorized into *Black-box/White-box* and *targeted/untargeted* attacks [144]. Black-box methods do not assume access to the trained weights of the model and only need the model output and confidence. White-box approaches on the other hand, require access to the model weights, architecture and gradients. Using targeted attacks, the user can specify a class into which the model should be fooled, while untargeted attacks only require the model to predict a different class for the adversarial example than for the original input. The perturbations are often not recognizable by humans and can lead to misclassifications with high confidences. This type of phenomenon is not directly related to the topic of uncertainty quantification but still poses a large risk in safety critical applications. Adversarial examples are exploiting a weakness in machine learning applications and especially deep neural networks due to their large number of parameters and their decision boundaries extending to infinity [51], hence opening up a lot of attack vectors. Still, certain methods for uncertainty quantification might be able to detect these attacks and therefore make the real-world application of NNs more safe. Szegedy et al. [128] were the first to discover adversarial examples and formulated the problem of generating adversarial examples as a box-constrained optimization problem

$$\begin{aligned} \min_{\boldsymbol{\eta}} \quad & \|\boldsymbol{\eta}\|_2 \\ \text{s.t.} \quad & \operatorname*{arg\,max}_{k=1,\dots,K} \hat{p}(y = k \mid \boldsymbol{x} + \boldsymbol{\eta}) = y' \\ & \operatorname*{arg\,max}_{k=1,\dots,K} \hat{p}(y = k \mid \boldsymbol{x}) = y^{\dagger} \\ & y^{\dagger} \neq y' \\ & \boldsymbol{x} + \boldsymbol{\eta} \in [0,1]^d \end{aligned} \quad . \tag{4.18}$$

To be able to approximate solutions for Eq. (4.18), Szegedy et al. [128] reformulate it in

$$\begin{aligned} \min_{\boldsymbol{\eta}} \quad & c\|\boldsymbol{\eta}\|_2 + J_{\boldsymbol{\theta}}(\boldsymbol{x}, y') \\ \text{s.t.} \quad & \boldsymbol{x} + \boldsymbol{\eta} \in [0,1]^d \end{aligned} \quad . \tag{4.19}$$

Here, $J_{\boldsymbol{\theta}}(\boldsymbol{x}, y')$ is an error function (in the case of classification this could be, e.g., the cross-entropy loss), $\boldsymbol{x}$ the input, and y' the target label. They then apply a Limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS)[1] algorithm to approximate solutions for Eq. (4.19). The hyperparameter $c > 0$ is determined by performing line-search, which is what makes this approach slow in practice. Goodfellow, Shlens, and Szegedy [43] substitute the slow optimization of Eq. (4.19) by using backpropagation to perform a gradient update

$$\boldsymbol{\eta} = \epsilon \cdot \operatorname{sign}\left(\nabla_{\boldsymbol{x}} J_{\boldsymbol{\theta}}(\boldsymbol{x}, y')\right) , \tag{4.20}$$

which they call Fast Gradient Sign Method (FGSM). A single gradient update is enough, which makes this method suitable to generate adversarial examples on the fly to perform adversarial training, which can improve a DNN's ability to detect such attacks [88, 128]. Following publications further worked on the improvement of the FGSM method, e.g., by considering the raw gradient instead of the sign [109], adding a momentum term [30], or additional randomness [133]. Iterative variants have been explored in multiple works (e.g., [73, 88]) and proved to provide higher success rates with less perturbation. Kurakin, Goodfellow, and Bengio [73] use a smaller step size and clip the adversarial image in each iteration so that it is in an L_{p} ball around the original image, which they call Basic Iterative Method (BIM). Additionally, they show that it is also possible to generate adversarial examples

1 This is a quasi-Newton algorithm for unconstrained non-linear optimization problems, approximating the original algorithm by Broyden, Fletcher, Goldfarb and Shanno with a limited amount of memory [84].

that fool the classification model to a specific class. In their setup they use the least likely class

$$y^{\mathrm{LL}} = \underset{k=1,\dots,K}{\arg\min}\, \hat{p}(y = k \mid \boldsymbol{x}), \tag{4.21}$$

which they call Iterative Least Likely Class (ILLC) method. Empirically the ILLC has a higher impact on the target's classification accuracy in a white-box setup. Besides this, they also demonstrate that it is possible to craft physical adversarial examples by printing them on a piece of paper. Even though the process of printing and capturing introduces a transformation that reduces the effectiveness of adversarial examples, these attacks also work in a black-box scenario, which causes a large threat for real-world applications. Madry et al. [88] do not apply clipping onto the adversarial image but rather project it onto the ϵ ball around the original image after each iteration. This method is called Projected Gradient Descent (PGD) and Madry et al. [88] use it to train their model to be resistant to these type of attacks. In their work they also claim that PGD might be a "universal" adversary in the class of all adversaries using first-order derivatives. Other iterative approaches include DeepFool [93] and the Carlini and Wagner Attack [15]. The DeepFool approach by Moosavi-Dezfooli, Fawzi, and Frossard [93] linearizes the classification model around an example $\boldsymbol{x}$ and tries to find the closest decision boundary, given the linearized gradient at $\boldsymbol{x}$. By approximating the closest decision boundary, the DeepFool attack generates smaller perturbations than the L-BFGS and FGSM methods but still has a higher runtime than the FGSM. Carlini and Wagner [15] explored different formulations of Eq. (4.18) by considering other loss functions $J_{\boldsymbol{\theta}}(\boldsymbol{x})$. In their work, they formulate the optimization problem as

$$\begin{aligned} \min_{\boldsymbol{\eta}} \quad & \|\boldsymbol{\eta}\|_{\mathrm{p}} + c \cdot J_{\boldsymbol{\theta}}(\boldsymbol{x} + \boldsymbol{\eta}) \\ \text{s.t.} \quad & \boldsymbol{x} + \boldsymbol{\eta} \in [0,1]^d \end{aligned}\,, \tag{4.22}$$

and conclude empirically that

$$J_{\boldsymbol{\theta}}(\boldsymbol{x}) = \max\left(\max_{y' \in \mathcal{Y} \setminus \{y^{\mathrm{MAP}}\}} \left(\boldsymbol{f}(\boldsymbol{x} \mid \boldsymbol{\theta})_{y'}\right) - \boldsymbol{f}(\boldsymbol{x} \mid \boldsymbol{\theta})_{y^{\mathrm{MAP}}}, 0 \right), \tag{4.23}$$

works best. Here, $\boldsymbol{f}(\boldsymbol{x} \mid \boldsymbol{\theta})$ are the logits of the classification model as used in Chapter 3 and $y^{\mathrm{MAP}} = \arg\max_{k=1,\dots,K} \hat{p}(y = k \mid \boldsymbol{x})$ is the predicted class of the original input $\boldsymbol{x}$. Carlini and Wagner [15] circumvented the box-constraint $\boldsymbol{x} + \boldsymbol{\eta} \in [0,1]^d$ by a reparameterization of $\boldsymbol{\eta}$ into

$$\boldsymbol{\eta} = \frac{1}{2}\left(\tanh(\omega) + 1\right) - \boldsymbol{x}\,. \tag{4.24}$$

This forces the resulting adversary to be in the interval $[0,1]$ and can be seen as a smooth approximation of a clipping operation [15]. Due to the fact that Eq. (4.24) is differentiable, Carlini and Wagner [15] used the Adam optimizer [64] to minimize the objective in Eq. (4.22). Similar to Szegedy et al. [128], the constant c is chosen by performing a binary search until they find a minimal c which satisfies $J_{\boldsymbol{\theta}}(\boldsymbol{x} + \boldsymbol{\eta}^*) \leq 0$, with $\boldsymbol{\eta}^*$ being the final solution of an optimization run.

The possibility of physical adversarial examples has also been explored by Sharif et al. [121]. In their work they craft adversarial examples for face detection models, demonstrating that it is possible to impersonate any person the model knows by wearing adversarially crafted glasses. To achieve this, they incorporate additional auxiliary loss functions to improve the smoothness and printability of the resulting adversarial examples. In total, the optimization problem for impersonating a person y_t is defined as

$$\min_{\boldsymbol{\eta}} - \log\left(\hat{p}(y = y_t \mid \boldsymbol{x} + \boldsymbol{\eta})\right) + \lambda_1 \cdot \mathrm{TV}(\boldsymbol{\eta}) + \lambda_2 \cdot \mathrm{NPS}(\boldsymbol{\eta})\,. \tag{4.25}$$

With $\mathrm{TV}(\boldsymbol{\eta})$ and $\mathrm{NPS}(\boldsymbol{\eta})$ being the *total variation* and *non-printability* score, respectively, which are defined as

$$\mathrm{TV}(\boldsymbol{\eta}) = \sum_{i,j} \left((\boldsymbol{\eta}_{i,j} - \boldsymbol{\eta}_{i+1,j})^2 + (\boldsymbol{\eta}_{i,j} - \boldsymbol{\eta}_{i,j+1})^2\right)^{\frac{1}{2}}\,, \tag{4.26}$$

$$\mathrm{NPS}(\boldsymbol{\eta}) = \sum_{i,j} \prod_{\tilde{p} \in \mathcal{P}} \|\tilde{p} - \boldsymbol{\eta}_{i,j}\|_1\,. \tag{4.27}$$

Here, $\boldsymbol{\eta}_{i,j}$ is the pixel of the perturbation at position (i, j) and $\mathcal{P} \subset [0, 1]^3$ is a set of RGB colors that are printable with the printer at hand (also called *gamut*). Minimizing the total variation results in adversarial examples that have a smooth transition between pixel values, as typically found in natural images. The non-printability score ensures that optimizing Eq. (4.25) results in a perturbation that is realizable by the printer. Equation (4.25) was optimized iteratively by using the gradient descent algorithm. In contrast to Eqs. (4.19) and (4.22), a loss on the intensity of $\boldsymbol{\eta}$ is not required, as the goal is to successfully defeat an automatic face detection model and not to make the attack unrecognizable by humans.

4.6 uncertainty applications and related tasks

Uncertainty Quantification (UQ) methods are especially important in safety critical applications such as medical diagnosis, autonomous driving or financial fraud detection. Additionally, there are a number of ML research topics that directly depend on good uncertainty estimates. This section gives an overview of these related tasks.

Novelty and *anomaly detection* [53, 96, 116] are from an application perspective directly related to UQ. Novel objects can be seen as a synonym for OoD objects and can be detected by classic UQ methods. On the other hand, anomalies are considered to be ID but reside in low density areas of the input space. As such, they are located at the boundary of the training distribution and similar to OoD objects. These might be a result of a labeling error or faulty sensor measurements and can harm the performance of the classification model during inference.

Active learning is the task of selecting unlabeled data points for labeling, such that the knowledge that the classification model gains from this sample is maximized.

DNNs require a large dataset which can be very expensive to label, and the ability to reduce annotation costs is of great interest in many real-world applications. Especially in situations where only a few experts are able to label the data points reliably, e.g., medical images or satellite images from other planets, selecting examples which contain the most information is crucial. Choosing data points for which the classification model has a high predictive uncertainty (termed *uncertainty sampling* [81]) is an intuitive procedure and several works show the effectiveness of the approach (e.g., [17, 81, 97]). Therein, epistemic uncertainty sampling showed superior performance compared to total or aleatoric uncertainty sampling, underlining the importance of being able to separate different sources of uncertainty [97].

Domain adaptation is the task of transferring the knowledge of a classification model from one domain (source) to another one (target). Source and target domain do not necessarily share the same label set and the data distributions in the two domains may be shifted in relation to each other. As an example, we might be interested in transferring a semantic segmentation model trained on urban street scenes to rural areas. In this case, the environment looks different, the frequency of certain car models may change, and even new classes may emerge, e.g., cows. A first step is to detect if a domain shift is present or not. Many works on the task of domain adaptation build evaluation setups where it is actually clear that the target domain is shifted. In practical applications this might not always be apparent, as a distribution shift can be caused by various factors. UQ can help to detect domain shifts by observing the overall level of aleatoric and epistemic uncertainty. Distinguishing between aleatoric and epistemic uncertainty in a situation where a shifted domain as well as OoD objects may appear is unlikely to perform well and in this case monitoring of the total amount of uncertainty should be pursued. Besides for the detection of domain shifts, uncertainties have also been used in numerous ways for the task of domain adaptation (e.g., [45, 114, 140]).

Explainability is an important research area for black-box predictors such as DNNs. Works on this topic supply methods to complement the predictions of a classification model with human-understandable reasons that led to its final prediction. In order to be able to understand what a classification model has learned during training, looking solely at the outputs of the model is not sufficient. Application areas such as medical image analysis may require a human as a final decision maker and to better incorporate DNNs into the final decision, additional information about prediction uncertainties can be of high value. The distinction between aleatoric and epistemic uncertainty and thus knowing if the model considers the current sample as a hard case (high aleatoric uncertainty) or as an unknown case (high epistemic uncertainty) is helpful information in many practical applications. While current works on explainability try to find regions in the input image which are influencing the classification decision the most (e.g., [5, 6]), prediction uncertainties should be considered as additional knowledge for the decision-making process.

Adversarial example detection aims at detecting the attacks described in Section 4.5. As there could be malicious attempts to attack NNs in practical applications, especially the detection during the testing phase is of interest. Targeted attacks are trying to maximize the softmax output for a different class than the originally predicted one. As a result, this reduces the overall entropy and might lead to overconfident predictions. External methods which are only analyzing the predicted class distribution of an NN (like entropy or MCP) are likely to fail in this detection task, which was also pointed out by Smith and Gal [123]. In the very same work, the authors also argue that an epistemic uncertainty measure is likely to be a good choice for detecting adversarial examples. This conclusion is based on the hypothesis that adversarial examples are lying outside the manifold of natural images, where an NN is largely unconstrained. A similar explanation for the effectiveness of adversarial examples was brought up by Szegedy et al. [128], who argue that these examples might be rarely seen ones from low density regions of the image space (corresponding to high epistemic uncertainty). Although many countermeasures against adversarial examples were proposed [144, p. 13], there still exist attacks such as the Carlini & Wagner attack [15] that are resistant to most of them.

4.7 SUMMARY

Deep Neural Networks (DNNs) have become an important part in many real world applications, some of which are safety-critical. This led to the development and research of methods for monitoring DNNs and their predictions. Table 4.1 summarizes the most important advantages and disadvantages of the methods presented in the previous sections. *Frequentist* approaches are especially suitable for resource-constrained applications but are usually not able to distinguish between aleatoric and epistemic uncertainty. *Bayesian* methods are theoretically grounded and can distinguish between different sources of uncertainty. However, they require a large amount of computational resources, restrict the model space and are less suitable for deep network architectures. The class of *ensemble* techniques brings benefits similar to those of Bayesian methods and typically increases overall model accuracy. On the other hand, they require additional memory during the inference phase. The recently emerged field of *generative* approaches delivers state-of-the-art OoD detection performance without the need for additional auxiliary data. The downside is that training these models requires more computational resources and the network architectures and learning frameworks are complex.

Despite the progress of recent years, there is still a lot of room for improvement. Giving theoretical guarantees on the model performance and confidence in the presence of domain shifts and unknown objects remains an open problem. Many of the aforementioned methods have only been evaluated on relatively easy and low-dimensional benchmarks and struggle when used for higher dimensional data.

Method	Advantages	Disadvantages
Frequentists	• Efficient computation of confidence measures during inference • External approaches can be used with existing models • OvA approaches bring a number of benefits	• Internal methods require changing the network architecture or training scheme • Confidence measures are usually not able to distinguish between aleatoric and epistemic uncertainties • OvA approaches require more special handling for the loss function and the final aggregation • Often require auxiliary data representative for the OoD domain
Bayesian	• Theoretically grounded uncertainty • Separation into aleatoric and epistemic uncertainty is possible	• Training takes a large amount of computational resources • Inference takes considerably more computational resources • Restricted model space and unclear which prior to choose
Ensembles	• Separation into aleatoric and epistemic uncertainty is possible • Ensembling usually increases the overall model accuracy • Training can be done in parallel	• Large memory footprint • Additional inference cost scales linear with the ensemble size
Generative	• State-of-the-Art performance in OoD detection • No auxiliary data required	• Complex architecture • Increased computation cost during training

Table 4.1: Summary of advantages and disadvantages of different Uncertainty Quantification (UQ) methods.

5 GRADIENT METRICS

As DNNs have many parameters and are high dimensional, one of the largest sources of uncertainty in this type of learning algorithm is the model uncertainty. The model uncertainty is contributing to the overall epistemic uncertainty, as described in Section 2.2. Bayesian approaches offer a theoretical way of quantifying model uncertainty, as parameters are represented by distributions. However, a major downside of these methods is their increased computational cost during inference as well as the restriction of the hypothesis space caused by some selected approximation techniques. Being the largest contributor to the epistemic uncertainty, it is highly desirable to better quantify the predictive model uncertainty without increasing the inference cost. Moreover, it would be beneficial if this would not require a change in the network architecture or training procedure. One way of solving these deficiencies is by measuring the sensitivity of a model using weight gradients. During the training phase we use SGD to find local minima in the loss landscape formed by our model and the training data. Although we are only approximating the loss over the whole dataset, in the large limit we are converging to a local minimum within the parameter space. Figure 5.1 illustrates two local minima (dashed red lines), which might be the convergence points of our training algorithm. In the deployment phase of the model we can encounter unknown (OoD) inputs, which are not represented in the training dataset. A loss on these unknown examples is likely to be different from the one on the training set, resulting in a different optimization landscape and therefore also other local minima. Computing the slope of the shifted loss (caused by the OoD object) w.r.t. the model parameters (Fig. 5.1, ($\boldsymbol{\theta}^3$)), will most likely result in a larger gradient than for ID examples (Fig. 5.1, ($\boldsymbol{\theta}^1$)).

5.1 CROSS-ENTROPY GRADIENTS WITH MAP TARGETS

In a test environment we do not have access to the real labels of the inputs, and it therefore remains to be defined how we can compute a loss in this setting. Given an input $\boldsymbol{x}$ and the trained model $\hat{p}$ with parameters $\boldsymbol{\theta}$, we can use the MAP decision rule from Eq. (3.24) to predict a label $\hat{y}^{\mathrm{MAP}} = \arg\max_{k=1,\dots,K} \hat{p}(y = k \mid \boldsymbol{x}, \boldsymbol{\theta})$. This prediction can be used as a pseudo label to compute the gradient of the cross-entropy loss as

$$\boldsymbol{\nabla}^{\mathrm{ce}}_{\boldsymbol{\theta}}(\boldsymbol{x}) := -\boldsymbol{\nabla}_{\boldsymbol{\theta}} \log\left(\hat{p}(y = \hat{y}^{\mathrm{MAP}} \mid \boldsymbol{x}, \boldsymbol{\theta})\right) . \tag{5.1}$$

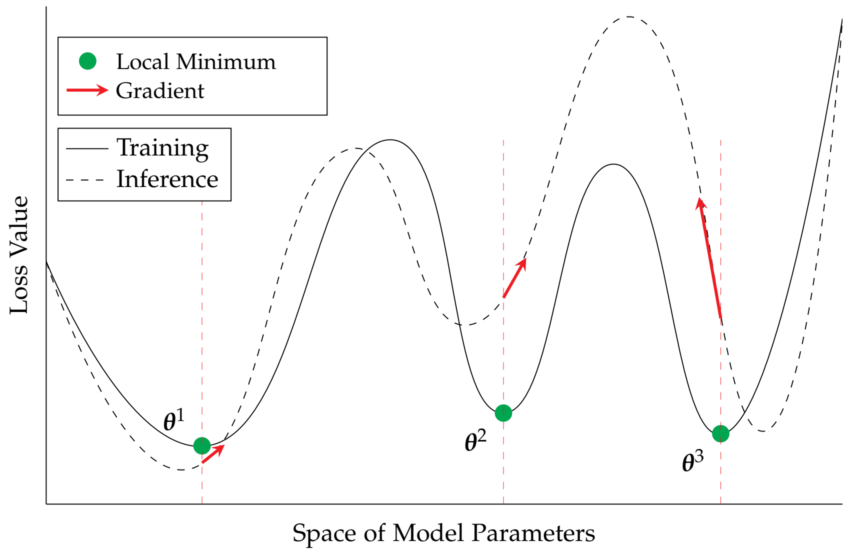

Figure 5.1: Illustration of gradient metrics. If an input is similar to the training data, it can be expected that the loss is similar to the one encountered during training. This results in a relatively small gradient ($\boldsymbol{\theta}^1$). On the other hand, if the input is unknown, it is likely that the loss will be greater and produces a larger gradient ($\boldsymbol{\theta}^3$). This graphic is inspired by Figure 7 of Gawlikowski et al. [38, p. 18].

As $\nabla^{\text{ce}}_{\boldsymbol{\theta}}(\boldsymbol{x})$ is a high dimensional gradient, we need to extract summarization statistics in order to get a scalar confidence value describing the epistemic uncertainty originating from the model parameters. This can be accomplished by applying different metrics such as

$$\text{(L_p-norm)} \qquad \|\nabla^{\text{ce}}_{\boldsymbol{\theta}}(\boldsymbol{x})\|_\text{p}\,, \tag{5.2}$$

$$\text{(Minimum)} \qquad \min_i \nabla^{\text{ce}}_{\boldsymbol{\theta}}(\boldsymbol{x})_i\,, \tag{5.3}$$

$$\text{(Maximum)} \qquad \max_i \nabla^{\text{ce}}_{\boldsymbol{\theta}}(\boldsymbol{x})_i\,, \tag{5.4}$$

$$\text{(Mean)} \qquad \overline{\nabla^{\text{ce}}_{\boldsymbol{\theta}}}(\boldsymbol{x}) = \frac{1}{d_{\boldsymbol{\theta}}} \sum_{i=1}^{d_{\boldsymbol{\theta}}} \nabla^{\text{ce}}_{\boldsymbol{\theta}}(\boldsymbol{x})_i\,, \tag{5.5}$$

$$\begin{pmatrix}\text{Standard}\\ \text{Deviation}\end{pmatrix} \qquad \sqrt{\frac{1}{d_{\boldsymbol{\theta}}} \sum_{i=1}^{d_{\boldsymbol{\theta}}} (\nabla^{\text{ce}}_{\boldsymbol{\theta}}(\boldsymbol{x})_i - \overline{\nabla^{\text{ce}}_{\boldsymbol{\theta}}}(\boldsymbol{x}))^2}\,. \tag{5.6}$$

$$\tag{5.7}$$

As OoD objects are going to still contain familiar features from the ID classes, not all parameters will produce a large gradient for these inputs. However, we can expect that unknown inputs contain underrepresented feature constellations, which will result in larger gradients for some parameters. This means that summarization metrics which are able to preserve magnitudes (L_p-norm, Minimum, Maximum) are likely to perform better as a confidence measure for detecting OoD inputs. Using this kind of epistemic uncertainty quantification for detecting OoD inputs was first proposed by Oberdiek, Rottmann, and Gottschalk [100], which the authors termed *gradient metrics*.

In Eq. (5.1) we compute the gradient over all network parameters. We can speed up the computation of gradient metrics significantly by only selecting a subset of weights. It has been shown in previous works that the early layers of a neural network learn low level features such as edges, while layers which are closer to the output learn more complex high level features [142, 145]. Low level features can be matched to many inputs regardless of the class, which is one of the reasons why pre-training, even on unrelated inputs, can be beneficial. On the other hand, high level features are representative for the training dataset and might not trigger for OoD inputs, which makes these layers suitable for distinguishing between ID and OoD examples. This also connects well to the backpropagation algorithm, which first propagates loss values to the final layers, making the computation of gradient metrics on these even more efficient.

While the gradient metrics can be used on their own as a measure for model uncertainty, combining multiple such metrics into a single score holds the possibility that the combined score might be a more informative uncertainty measure. Unfortunately, the aggregation of multiple metrics is not straight forward as they have different value ranges and interpretations regarding ID and OoD inputs. Training a meta classifier to do so offers the most direct approach. A meta classifier receives various gradient metrics as input and predicts a probability that the given input is ID. The model of the meta classifier can be any machine learning algorithm that is able to perform binary classification, such as SVMs, random forests, logistic regression or NNs. However, choosing suitable training data remains a problem for all these methods. As our goal is to quantify epistemic (model) uncertainty, we need representatives for both the ID and OoD domain. Choosing the right type of data for the OoD domain is not trivial and seems to be impossible considering that it is basically infinite. If we still want to attempt to collect an OoD dataset, we could gather a large set of samples which do not share any classes with the ID data (so-called *auxiliary* data [53]). This approach usually improves the overall OoD detection performance but comes at the cost of having to collect additional samples. Using Gaussian- and uniform noise provides a way of gathering free data as a proxy for OoD objects. On the other hand, this auxiliary data does not contain much variability and is far away from the training distribution, reducing its effectiveness.

Gradient metrics belong to the class of external frequentist approaches (see Section 4.1). This makes them suitable for resource constrained applications, and they can be applied to already trained models without altering the network architecture or requiring a re-training.

5.2 SIMILAR WORKS

Until now, gradient metrics have been employed in a number of similar works (e.g., [55, 60, 78]). The L_2-norm as a summarization statistic is also used by Huang, Geng, and Li [60] and Lee and AlRegib [78]. Lee and AlRegib [78] use *confounding labels* instead of the predicted class as a pseudo label to compute the gradients. Confounding labels differ from a classic one-hot encoding in that they can have multiple classes active. For a classification model with K classes, they define a confounding label $\boldsymbol{y}^{\text{cl}}$ as a vector of length K with c 1's and $c \in \{0, \dots, K\} \setminus \{1\}$. In their experiments they choose $c = K$, thus creating a pseudo label where each ID class is active and compute the binary cross entropy between the prediction and the pseudo label

$$\begin{aligned}
\boldsymbol{\nabla}^{\text{cl}}_{\boldsymbol{\theta}}(\boldsymbol{x}) :&= \nabla_{\boldsymbol{\theta}} \left[\frac{1}{K} \sum_{i=1}^{K} (\boldsymbol{y}^{\text{cl}}_i \cdot \log(\hat{\boldsymbol{y}}_i) + (1 - \boldsymbol{y}^{\text{cl}}_i) \cdot \log(1 - \hat{\boldsymbol{y}}_i)) \right] \\
&= \nabla_{\boldsymbol{\theta}} \left[\frac{1}{K} \sum_{i=1}^{K} \log(\hat{\boldsymbol{y}}_i) \right] \\
&= -\nabla_{\boldsymbol{\theta}} H(u, \hat{\boldsymbol{y}}) \,.
\end{aligned} \tag{5.8}$$

As can be seen in Eq. (5.8), this is equivalent to computing the negative cross-entropy loss between the prediction and a uniform class distribution.

The same approach is taken by Huang, Geng, and Li [60], where the authors compute the gradient on the KLD between a uniform class distribution and the model prediction

$$\boldsymbol{\nabla}^{\text{KL}}_{\boldsymbol{\theta}}(\boldsymbol{x}) := \nabla_{\boldsymbol{\theta}} \text{KL}(u \parallel \hat{\boldsymbol{y}}) \tag{5.9}$$
$$= \nabla_{\boldsymbol{\theta}} \left[H(u, \hat{\boldsymbol{y}}) - H(u) \right] \tag{5.10}$$
$$= \nabla_{\boldsymbol{\theta}} H(u, \hat{\boldsymbol{y}}) \tag{5.11}$$

Both approaches by Huang, Geng, and Li [60] and Lee and AlRegib [78] result in an average cross-entropy loss over all classes, which might gather more information compared to the cross-entropy only on the predicted class. Huang, Geng, and Li [60] also experiment with different summarization statistics but settle on the L_1-norm. They also find that computing the norm on the weights of the last layer is sufficient, speeding up the computation drastically. In addition to Lee and AlRegib [78], the authors of [60] additionally apply temperature scaling [47] to reduce the overconfidence of the network.

Hornauer and Belagiannis [55] use gradient metrics to quantify uncertainties on the task of monocular depth estimation. To achieve this, they compute depth maps on two instances of the same image, where one is augmented by horizontal flipping. On the resulting depth maps they compute an auxiliary loss from which they can gather gradients. They only consider gradients on a single intermediate feature representation and upsample these to the original image size. The major difference to Huang, Geng, and Li [60], Lee and AlRegib [78], and Oberdiek, Rottmann, and Gottschalk [100] is the application to a regression problem instead of classification.

6 UNCERTAINTY QUANTIFICATION GAN

In Chapter 5 we discussed a way to quantify model uncertainty based on the magnitude of parameter gradients. However, this method is only capturing model uncertainty and no data uncertainty, which is another major contributor to the epistemic uncertainty, as described in Section 2.2. Data uncertainty is a result of unknown input data, which is not covered by our training dataset. Because it is unknown, it is particularly difficult to describe this region as it is essentially the complement of our training dataset. The fact that the decision boundaries of neural networks can extend to infinity and can lead to arbitrarily high confidence values (see Fig. 3.2) also makes it impossible for many methods to describe data uncertainty.

Let us assume that we are able to train a reject classifier in such a way that we are forcing the DNN to place the rejection decision boundary at the boundary of our training dataset. Then we would be able to classify unknown inputs into the OoD domain and use the prediction confidence of the reject class as our proxy for epistemic uncertainty. A stepping stone on the way to this goal is the ability to gather data which "shields" or wraps around our training dataset and acts as a proxy for the OoD domain. Generative methods can be utilized to generate auxiliary data and in the works by Lee et al. [79], Ngo et al. [96], and Sensoy et al. [117] attempts were made to generate boundary samples. Covering the boundary of a distribution is not an easy task, especially in higher dimensional problems as, e.g., in the case of image data. To this end, analysis by Vernekar et al. [136] has shown that the approach by Lee et al. [79] is prone to mode collapse and does not generate boundary samples. While the method by Sensoy et al. [117] seems to not suffer from mode collapse, their toy examples show that the generator distribution is similar to the training samples with higher variance. In this case, the higher density within the training set is likely to superimpose the generator distribution, putting more emphasis on the generated OoD samples. This seems to be beneficial for the detection of OoD inputs but is still not equal to generating boundary samples. Ngo et al. [96] also proposed, independently of Lee et al. [79], to generate boundary samples using a modified GAN setup. The mentioned mode collapse is addressed by additionally including a *dispersion loss*, which penalizes the generator to place all samples on a single point in the boundary.

The GAN approach for boundary samples has a lot of potential, as shown by the previously cited works. There still remains a lot of room for improvement, which we will address by the following architectural and theoretical changes

1. *Dimensionality Reduction*
   In high dimensions, the space becomes sparse, given the same number of samples. In very high dimensions distance metrics and the notion of nearest neighbors becomes meaningless [2]. Because of the *curse of dimensionality*, learning to generate boundary samples becomes extremely difficult in high dimensions. We can counteract this to a certain degree by training the GAN within a lower-dimensional latent space. To transform the input samples into a low-dimensional space we use an Autoencoder (AE), which is a powerful representation learning tool.

2. *Class Conditioning*
   Another way of reducing the complexity of the boundary is by conditioning on the classes. Assuming class regions in the low-dimensional space to be somewhat compact, covering the boundary of a single class might be easier to achieve than for the whole training distribution. Additionally, the class conditioning can be utilized by an OvA classifier. This can be combined with a conditional Autoencoder (cAE) to also introduce the class-conditioning into the latent space and to be able to decode latent embeddings given a class label. The generator is then producing Out-of-Class (OoC) auxiliary data instead of Out-of-Distribution (OoD).

3. *One-versus-All* (OvA) *Classifier*
   When generating OoC data, it might happen that we are generating ID data for other classes, as we can not guarantee that OoC data will also be OoD. We can not handle this problem with a standard all-vs-all softmax classifier, as it does not have the notion of "does not belong to this class" and "not belonging to any known class". In fact, this problem can be an advantage if we choose an OvA classifier. Using this model, we are obtaining free auxiliary OoC data, as all other classes also belong to the OoC domain. By combining the training data with the generated auxiliary data, we can achieve a better boundary coverage.

4. *Low-Dimensional Regularization*
   Ngo et al. [96] are employing a dispersion loss to fight the mode collapse of the generator. They define their dispersion loss by computing an L_2 loss within the original learning domain $\mathcal{X}$. The curse of dimensionality greatly affects many distance metrics as the density becomes less, increasing the overall distances between points [2]. This makes it hard to balance the loss with other loss components during training, and hyperparameters might not be generally applicable across different datasets. Instead, it is a good idea to define such a regularizer on our low-dimensional latent space. By utilizing other distance measures we can also avoid the value range shift for different number of dimensions.

In the following sections we will discuss in more detail how the proposed changes from above can be implemented.

6.1 ONE-VS-ALL CLASSIFIER

The Uncertainty Quantification GAN (UQGAN) framework utilizes an OvA classifier to form class predictions and associated uncertainty measures. Given a training dataset $\mathcal{S} \subseteq \mathcal{X} \times \mathcal{Y}$ with relative class frequencies $\hat{p}(y) = \frac{|\{y'=y|(\boldsymbol{x},y')\in\mathcal{S}\}|}{|\mathcal{S}|}$, we define $C(i \mid \boldsymbol{x}, y)$ as the probability of sample $(\boldsymbol{x}, y) \in \mathcal{X} \times \mathcal{Y}$ being ID and $C(o \mid \boldsymbol{x}, y) = 1 - C(i \mid \boldsymbol{x}, y)$ as the one of being OoD, respectively. The classifiers can be modeled with an ensemble of independent NNs but the preferred way is to construct them as a single NN with $K \geq 2$ sigmoid outputs, modeling all $C(i \mid \cdot, y),\ y \in \mathcal{Y}$. This way the number of parameters is reduced by a factor of K and the learned features can be shared across classes, lowering the risk of overfitting. Note that this setup does not differ from a classic softmax based NN architecture, only the softmax activation is replaced by sigmoid activations. During training, samples corresponding to a class y are serving as in-class data and all others as OoC data. A benefit with this model is that we are already obtaining free auxiliary data for the OoC part. The training objective of the OvA classifier is then given by a weighted empirical cross entropy, similar to the multi-class setup

$$\min_C \frac{1}{|\mathcal{S}|} \sum_{(\boldsymbol{x},y)\in\mathcal{S}} \left[-\log(C(i \mid \boldsymbol{x}, y)) - \frac{1}{K-1} \sum_{y'\in\mathcal{Y}\setminus\{y\}} \frac{\hat{p}(y)}{\hat{p}(y')} \log(C(o \mid \boldsymbol{x}, y')) \right] \tag{6.1}$$

Note that instead of writing $\min_{\boldsymbol{\theta}_C}$ and $C(i \mid \boldsymbol{x}, y, \boldsymbol{\theta}_C)$ we are leaving out the dependence on the parameter set $\boldsymbol{\theta}_C$ and write $\min_C$ for brevity. In Eq. (6.1) the factor $\frac{1}{K-1}$ is balancing the in-class and the OoC loss. Suppose all classes are equally frequent in S, meaning $\hat{p}(y) = \frac{1}{K},\ \forall y \in \mathcal{Y}$. Then, for each class y, there are $K-1$ times more data samples belonging to the OoC than the in-class regime. This imbalance needs to be accounted for to avoid overfitting to the OoC data and also assures that our learned classifier is converging to the desired quantity, as can be seen later. Now, assume the frequency of class samples in $\mathcal{S}$ is not uniform. This case also poses the risk of overfitting to the most frequent class. This is accounted for using the class dependent weighting of $\frac{\hat{p}(y)}{\hat{p}(y')}$ in Eq. (6.1). Assume, without loss of generality, that we have a constant in-class loss over all classes, thus $-\log(C(i \mid \boldsymbol{x}, y)) = 1,\ \forall y \in \mathcal{Y}$. A single class $y \in \mathcal{Y}$ contributes a portion of $\hat{p}(y)$ to its own in-class loss. We can then scale the individual in-class losses of each class by a factor of $\frac{1}{\hat{p}(y)}$ to ensure a balanced loss contribution of each class. The same factor applies for the OoC loss part for the respective class, which leaves us with

$$\min_C \frac{1}{|\mathcal{S}|} \sum_{(\boldsymbol{x},y)\in\mathcal{S}} \left[-\frac{1}{\hat{p}(y)} \log(C(i \mid \boldsymbol{x}, y)) - \frac{1}{K-1} \sum_{y'\in\mathcal{Y}\setminus\{y\}} \frac{1}{\hat{p}(y')} \log(C(o \mid \boldsymbol{x}, y')) \right] \tag{6.2}$$

As the individual classifiers $C(i \mid \cdot, y)$ do not depend on other classes, we can multiply by a factor of $\hat{p}(y)$ to arrive at Eq. (6.1). To be able to perform multi-class predictions, we are aiming to find an estimator $\hat{p}(y \mid \boldsymbol{x})$ which converges to the true distribution $p(y \mid \boldsymbol{x})$ for $|\mathcal{S}| \to \infty$. This can be achieved by applying the transformation

$$\tilde{C}(i \mid \boldsymbol{x}, y) = \frac{\frac{1}{K}C(i \mid \boldsymbol{x}, y)}{\frac{1}{K}C(i \mid \boldsymbol{x}, y) + \frac{K-1}{K}C(o \mid \boldsymbol{x}, y)}, \tag{6.3}$$

and finally aggregating the individual ID probabilities as in the following

$$\hat{p}(y \mid \boldsymbol{x}) = \frac{\tilde{C}(i \mid \boldsymbol{x}, y)\hat{p}(y)}{\sum_{y' \in \mathcal{Y}} \tilde{C}(i \mid \boldsymbol{x}, y')\hat{p}(y')}. \tag{6.4}$$

Consider the following typical assumptions of statistical learning theory (see Section 2.1)

Assumption 1. *We make the following assumptions for our OvA classifiers C*

1. *We sample $\mathcal{S} \sim p(\boldsymbol{x}, y)$ i.i.d. and there is no GAN-generated data involved.*
2. *We assume, there exists a $C^* \in \mathcal{H}$ such that $\hat{p}(y \mid \boldsymbol{x}) = p(y \mid \boldsymbol{x})$, i.e., $p(y|\boldsymbol{x})$ is realizable. This is a realistic assumption for DNNs (see Section 2.1).*
3. *We can compute an empirical risk minimizer, i.e., we can determine a $C_{\mathcal{S}} \in \mathcal{H}$ which minimizes Eq.* (6.1) *for a given sample $\mathcal{S}$.*

Then we can state the following

Lemma 1 (Class Posterior). *Under Assumption 1 and training C on Eq.* (6.1) *it holds that*

$$\hat{p}(y \mid \boldsymbol{x}) = \frac{\tilde{C}(i \mid \boldsymbol{x}, y)\hat{p}(y)}{\sum_{y' \in \mathcal{Y}} \tilde{C}(i \mid \boldsymbol{x}, y')\hat{p}(y')} \xrightarrow{|\mathcal{S}| \to \infty} p(y \mid \boldsymbol{x})$$

Proof. Without loss of generality, consider a single OvA classifier $C(i \mid \boldsymbol{x}, y^*)$ with $y^* \in \mathcal{Y}$ fixed and define a counter-part class $\bar{y}^*$ (class "not y^*"), which contains all classes in $\mathcal{Y} \setminus \{y^*\}$. If we subsample our training dataset $\mathcal{S}$ as $\mathcal{S}_{y^*} \sim \tilde{p}_{y^*}(\boldsymbol{x}, y) = p(\boldsymbol{x} \mid y)\tilde{p}_{y^*}(y)$, with $\tilde{p}_{y^*}(y^*) = \frac{1}{2}$ and $\tilde{p}_{y^*}(y) = \frac{1}{2(K-1)}$, $\forall y \in \mathcal{Y} \setminus \{y^*\}$, we are weighting y^* and $\bar{y}^*$ equally, because $\sum_{y' \in \mathcal{Y} \setminus \{y^*\}} \tilde{p}_{y^*}(y') = \frac{1}{2} = \tilde{p}_{y^*}(y^*)$. The contribution of the single OvA classifier $C(i \mid \boldsymbol{x}, y^*)$ within Eq. (6.1) can then be summarized as

$$\frac{1}{|\mathcal{S}_{y^*}|} \sum_{(\boldsymbol{x}, y) \in \mathcal{S}_{y^*}} \left[-\mathbb{1}_{\{y = y^*\}} \log(C(i \mid \boldsymbol{x}, y^*)) - \frac{1}{K-1} \mathbb{1}_{\{y \neq y^*\}} \frac{\frac{1}{2}}{\frac{1}{2(K-1)}} \log(C(o \mid \boldsymbol{x}, y^*)) \right] \tag{6.5}$$

$$= \frac{1}{|\mathcal{S}_{y^*}|} \sum_{(\boldsymbol{x}, y) \in \mathcal{S}_{y^*}} \left[-\mathbb{1}_{\{y = y^*\}} \log(C(i \mid \boldsymbol{x}, y^*)) - \mathbb{1}_{\{y \neq y^*\}} \log(C(o \mid \boldsymbol{x}, y^*)) \right], \tag{6.6}$$

which is the binary cross entropy loss for equal class weights of y^* and $\bar{y}^*$ (see Eq. (3.23) when choosing a Bernoulli distribution as data model assumption). As we have shown that Eq. (6.1) is yielding a balanced OvA classifier, we can follow that for $|\mathcal{S}| \to \infty$ we obtain

$$C(i \mid \boldsymbol{x}, y) \longrightarrow p(y \mid \boldsymbol{x}) = \frac{p(\boldsymbol{x} \mid y)p_U(y)}{p(\boldsymbol{x} \mid y)p_U(y) + p(\boldsymbol{x} \mid \bar{y})p_U(\bar{y})} \tag{6.7}$$

$$= \frac{p(\boldsymbol{x} \mid y)}{p(\boldsymbol{x} \mid y) + p(\boldsymbol{x} \mid \bar{y})}, \quad \forall y \in \mathcal{Y}, \tag{6.8}$$

in the sub-sampled OvA scenario. Let

$$p_U(\boldsymbol{x}, y) = p(\boldsymbol{x} \mid y)p_U(y) \tag{6.9}$$

be defined as the joint distribution of $\boldsymbol{x}$ and y w.r.t. a uniform class distribution $p_U(y) = \frac{1}{K}$, $\forall y \in \mathcal{Y}$. By re-weighting the OvA classifier with Eq. (6.3) we obtain

$$\tilde{C}(i \mid \boldsymbol{x}, y) = \frac{\frac{1}{K}C(i \mid \boldsymbol{x}, y)}{\frac{1}{K}C(i \mid \boldsymbol{x}, y) + \frac{K-1}{K}C(o \mid \boldsymbol{x}, y)} \tag{6.10}$$

$$\longrightarrow \frac{\frac{\frac{1}{K}p(\boldsymbol{x}|y)}{p(\boldsymbol{x}|y)+p(\boldsymbol{x}|\bar{y})}}{\frac{\frac{1}{K}p(\boldsymbol{x}|y)+\frac{K-1}{K}p(\boldsymbol{x}|\bar{y})}{p(\boldsymbol{x}|y)+p(\boldsymbol{x}|\bar{y})}} \tag{6.11}$$

$$= \frac{\frac{1}{K}p(\boldsymbol{x} \mid y)}{\frac{1}{K}p(\boldsymbol{x} \mid y) + \frac{K-1}{K}p(\boldsymbol{x} \mid \bar{y})} \tag{6.12}$$

$$= \frac{p(\boldsymbol{x} \mid y)p_U(y)}{p(\boldsymbol{x} \mid y)p_U(y) + p(\boldsymbol{x} \mid \bar{y})p_U(\bar{y})} \tag{6.13}$$

$$= \frac{p(\boldsymbol{x} \mid y)p_U(y)}{p_U(\boldsymbol{x})} \tag{6.14}$$

$$= p_U(y \mid \boldsymbol{x}) . \tag{6.15}$$

Finally, knowing the convergence to Eq. (6.15), we can conclude for $|\mathcal{S}| \to \infty$ and the re-weighted classifier $\tilde{C}$

$$\frac{\tilde{C}(i \mid \boldsymbol{x}, y)\hat{p}(y)}{\sum_{y'} \tilde{C}(i \mid \boldsymbol{x}, y')\hat{p}(y')} \longrightarrow \frac{p_U(y \mid \boldsymbol{x})p(y)}{\sum_{y'} p_U(y' \mid \boldsymbol{x})p(y')} \tag{6.16}$$

$$= \frac{\frac{p_U(\boldsymbol{x}|y)p_U(y)p(y)}{p_U(\boldsymbol{x})}}{\sum_{y'} \frac{p_U(\boldsymbol{x}|y')p_U(y')p(y')}{p_U(\boldsymbol{x})}} \tag{6.17}$$

$$= \frac{p(\boldsymbol{x} \mid y)p(y)}{\sum_{y'} p(\boldsymbol{x} \mid y')p(y')} \tag{6.18}$$

$$= p(y \mid \boldsymbol{x}) . \tag{6.19}$$

□

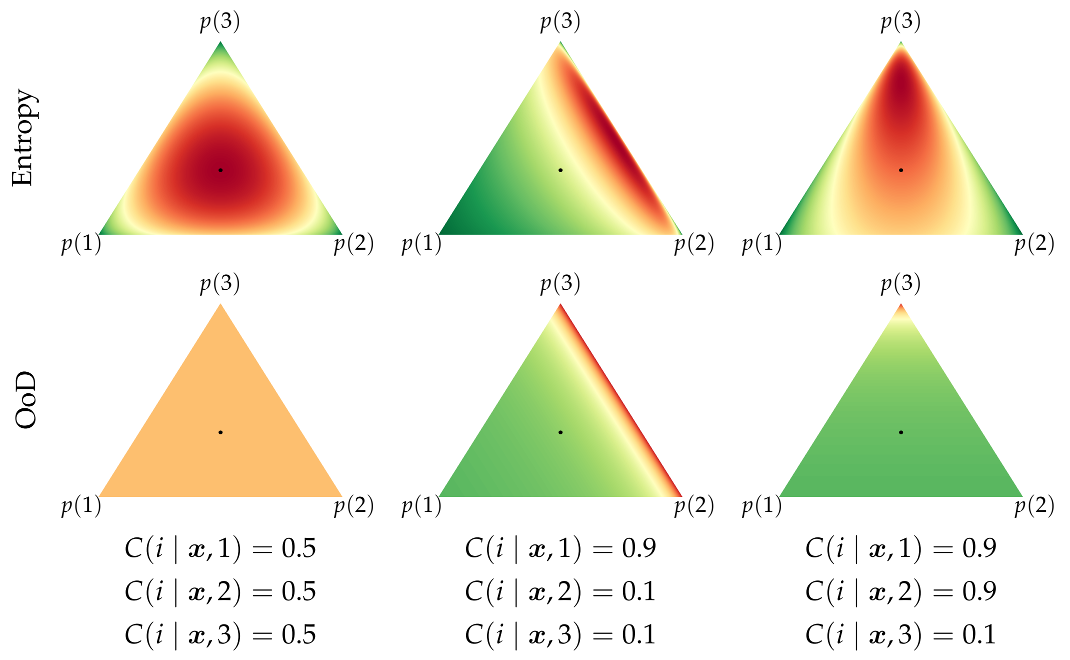

Figure 6.1: 2-simplexes over $\hat{p}(y)$ of entropy ($\tilde{H}(\hat{p}(y \mid \boldsymbol{x}))$) and OoD probability ($\tilde{C}(o \mid \boldsymbol{x})$) values. Top row: entropy, Bottom row: OoD probability. Green corresponds to low values and red to high ones. Columns are showing different combinations of predicted conditional ID probability. The black dot is showing the center of a simplex (point of uniform class frequency).

6.1.1 *One-vs-All Uncertainties*

After we defined the theoretical framework for our OvA classifier it remains to be defined how we can compute uncertainties. By using this type of classification model we can naturally encode the idea of "not belonging to this class", which includes OoD inputs. This is not possible with a softmax based classifier, which is trained with a closed-world assumption. As Lemma 1 provides us with the proof that $\hat{p}(y \mid \boldsymbol{x})$ is approximating the underlying class posterior $p(y \mid \boldsymbol{x})$, we can compute the aleatoric uncertainty associated with a prediction using the (normalized) entropy (see Eq. (2.13))

$$\tilde{H}\left(\hat{p}(y \mid \boldsymbol{x})\right) = -\frac{1}{\log(K)} \sum_{k=1}^{K} \hat{p}(y = k \mid \boldsymbol{x}) \log(\hat{p}(y = k \mid \boldsymbol{x}))\,. \tag{6.20}$$

For the epistemic uncertainty we conveniently get in-class probabilities for each class via $C(i \mid \cdot, y)$ and OoC probabilities as $C(o \mid \cdot, y) = 1 - C(i \mid \cdot, y)$. However, we need a way to combine the ensemble of binary classifiers into a single one to form a final epistemic uncertainty score. Usually, this is done by taking a maximum confidence approach $C(i \mid \boldsymbol{x}) = \max_{y \in \mathcal{Y}} C(i \mid \boldsymbol{x}, y)$, which is commonly used for multi-class SVMs [8, pp. 338–339]. However, this has the problem that the confidence levels of the individual ensemble members might not be equal due to imbalanced data settings or easier to distinguish samples, which introduces a bias into this decision rule. By using the result from Lemma 1 we can compute

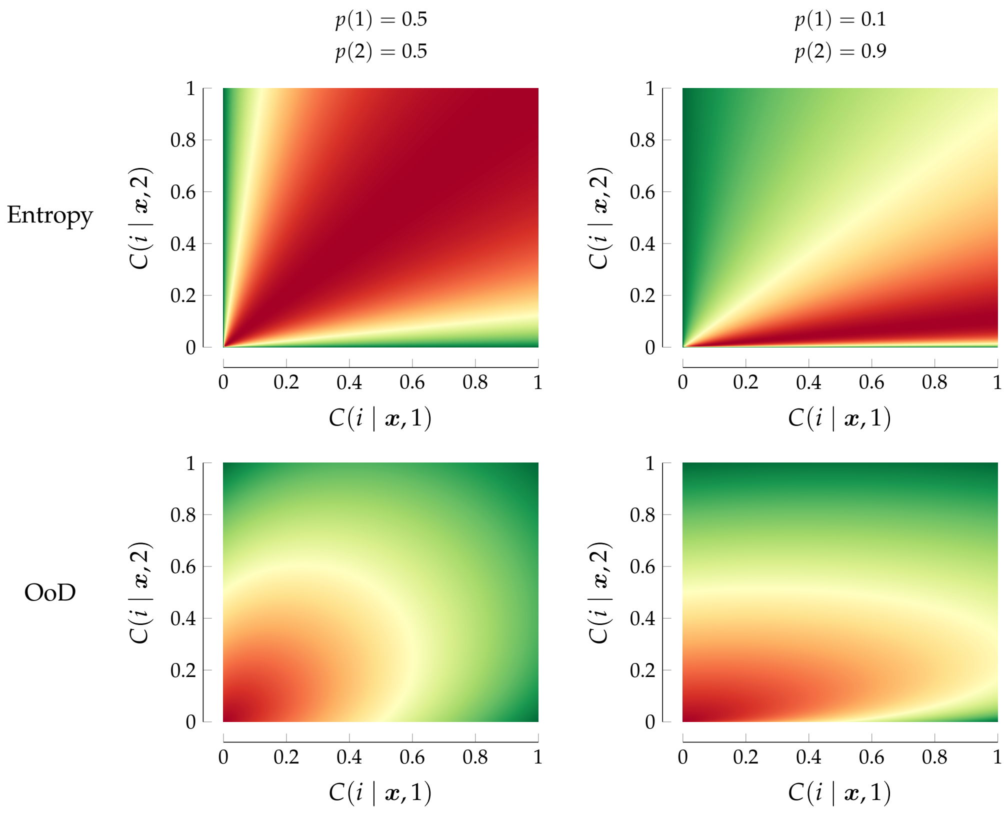

Figure 6.2: Entropy and OoD probability for different predicted in-class probabilities and fixed class frequencies. Top row: entropy, Bottom row: OoD probability. Green corresponds to low values and red to high ones. First column is showing a uniform class frequency while the second column displays a class imbalance.

a theoretically grounded fusion of all individual in-class probabilities to form a global ID probability

$$\tilde{C}(i \mid \boldsymbol{x}) = \sum_{k=1}^{K} \tilde{C}(i \mid \boldsymbol{x}, k)\hat{p}(y = k \mid \boldsymbol{x}) \overset{(6.4)}{=} \sum_{k=1}^{K} \frac{\tilde{C}(i \mid \boldsymbol{x}, k)^2 \hat{p}(k)}{\sum_{k'=1}^{K} \tilde{C}(i \mid \boldsymbol{x}, k')\hat{p}(k')}, \tag{6.21}$$

$$\tilde{C}(o \mid \boldsymbol{x}) = 1 - \tilde{C}(i \mid \boldsymbol{x}). \tag{6.22}$$

In Eq. (6.21) we weight each $\tilde{C}(i \mid \boldsymbol{x}, k)$ by its class posterior $\hat{p}(y = k \mid \boldsymbol{x})$. This is similar to the work of Franchi et al. [35] but instead of training an additional all-vs-all classifier, the weighting is integrated into the ensemble of binary classifiers. It is interesting to observe, that in the case of uniform in-class probability, thus $\tilde{C}(i \mid \boldsymbol{x}, k) = c, \forall k = 1, \ldots, K, c \in [0, 1]$, we get

$$\tilde{C}(i \mid \boldsymbol{x}) = \sum_{k=1}^{K} \frac{c^2 \hat{p}(y)}{\sum_{k'=1}^{K} c\hat{p}(y')} = c \cdot \sum_{k=1}^{K} \frac{\hat{p}(k)}{\sum_{k'=1}^{K} \hat{p}(k')} = c\,. \tag{6.23}$$

Which means, that the overall ID probability is no longer depending on the class-frequency but only on the level of in-class confidence. This can also be observed in Fig. 6.1 (second row, first column). In contrast to this, the aleatoric uncertainty quantified by the (normalized) entropy over Eq. (6.4) is highly dependent on the class frequency, which can also be seen in Fig. 6.1 (top row). The first column corresponds to a prediction of a uniform in-class probability for each class, resulting in high entropy values, only reducing in a setting with a major imbalance to a single class in the data. When a single OvA classifier is predicting a high in-class probability, we only get a high entropy if the same class has a very low frequency relative to the other two. In the last column, which shows a high in-class probability for two of the three classes, we can see a similar setting to the first column but slightly shifted, only expressing low entropy for an extreme imbalance in the data, decaying faster for the two classes with a high in-class confidence. Similar observations can be made in Fig. 6.2, which also shows entropy and OoD probability, but keeps the class frequency fixed while varying the in-class probabilities. The two-class setting shows that for a balanced class frequency we get a linear relationship between $C(i \mid \boldsymbol{x}, 1)$ and $C(i \mid \boldsymbol{x}, 2)$ when considering maximum entropy (as derived above).

Note that when implementing the class posterior from Eq. (6.4), an $0 < \epsilon \ll 1$ needs to be added to each $\tilde{C}(i \mid \boldsymbol{x}, y)$ in order to avoid a division by zero and still receive a valid class posterior. For data far away from the training distribution we can expect that all $\tilde{C}(i \mid \boldsymbol{x}, y)$ are zero, resulting in $\tilde{C}(i \mid \boldsymbol{x}) = \epsilon$, as derived above. If the classifier is forced to predict a class label for such an input, the class with the highest frequency in the training dataset will be chosen. To take the class with the highest occurrence frequency sounds reasonable for an input on which we have no prior knowledge. At the same time, the model has a prediction entropy of

$$\tilde{H}(\hat{p}(y \mid \boldsymbol{x})) = \frac{1}{\log(K)} \sum_{y' \in \mathcal{Y}} \hat{p}(y') \log(\hat{p}(y')) \,. \tag{6.24}$$

This means that the entropy of the class posterior on OoD data is equal to the entropy of the class distribution in the training dataset, which is 1 in the case of uniform class frequency. Although the input is far away from the training data, we have a large aleatoric uncertainty for the prediction. Having a high data uncertainty for such inputs is counter-intuitive, and to solve these situations we need to consider epistemic and aleatoric uncertainty jointly in this framework. We first need to detect OoD inputs based on the level of epistemic uncertainty before we can consider the aleatoric uncertainty. In a practical scenario this makes sense, as we can not interpret classification results on OoD data in a meaningful way. Also, this procedure is supported by the Bayesian predictive model assumption in Eq. (2.16), which formulates a chain of dependencies, where model uncertainty is influencing distribution uncertainty and this in turn is influencing data uncertainty.

The current OvA theoretical framework is able to separate aleatoric and epistemic uncertainty. However, since we are still learning based on maximum likelihood, resulting in a single weight configuration, the model uncertainty is not included. As model uncertainty is in the case of DNNs especially important to consider, we can apply Monte Carlo-Dropout (MCD) to our OvA classifier. MCD does not require a change in the model architecture or training procedure and is thus a good candidate to improve the quality of epistemic uncertainty estimates. Besides that, we are also able to improve the quantification of aleatoric uncertainty and the prediction accuracy as the data uncertainty is also influenced by the model uncertainty (see Section 2.2). This comes at the cost of additional computational effort, however. When given a set of T model weights $\{\boldsymbol{\theta}^t\}_{t=1}^{T}$, we include model uncertainty by computing

$$\hat{p}^T(y \mid \boldsymbol{x}) = \frac{1}{T}\sum_{t=1}^{T} \hat{p}(y \mid \boldsymbol{x}, \boldsymbol{\theta}^t)\,, \tag{6.25}$$

$$\tilde{C}^T(i \mid \boldsymbol{x}) = \frac{1}{T}\sum_{t=1}^{T} \tilde{C}(i \mid \boldsymbol{x}, \boldsymbol{\theta}^t)\,. \tag{6.26}$$

Aleatoric uncertainty is then given by the normalized entropy over $\hat{p}^T(y \mid \boldsymbol{x})$ and epistemic uncertainty as usual by $\tilde{C}^T(o \mid \boldsymbol{x}) = 1 - \tilde{C}^T(i \mid \boldsymbol{x})$. We will reference the model without MCD as UQGAN and with it as Uncertainty Quantification GAN - Monte Carlo Dropout (UQGAN-MCD).

6.2 CONDITIONAL GENERATIVE ADVERSARIAL NETWORK

In the preceding section 6.1 we derived the theory of our OvA classifier. This section will describe how to integrate the classification model into a conditional Generative Adversarial Network (cGAN) architecture to be able to generate class-conditional boundary samples. As described in the introduction to this chapter, we want to reduce the dimensionality of the learning problem by training a cAE on our input data $\mathcal{X}$. For this we will utilize an autoencoder which receives the class information to each example as an additional input to the encoder as well as the decoder part. Notation wise we will call $A(\boldsymbol{x}, y)$ as the complete cAE pipeline, $A_{\text{enc}}(\boldsymbol{x}, y)$ will be the encoder part, and $A_{\text{dec}}(\boldsymbol{z}, y)$ the decoder part with $\boldsymbol{z}$ being a latent vector. The cAE is trained with a pixel-wise binary cross-entropy as the reconstruction error, thus the total optimization objective is

$$\min_A \frac{1}{|\mathcal{S}|} \sum_{(\boldsymbol{x},y)\in\mathcal{S}} \left[\frac{1}{N_{\boldsymbol{x}}} \sum_{i=1}^{N_{\boldsymbol{x}}} \left[\boldsymbol{x}_i \cdot \log(\hat{\boldsymbol{x}}_i) + (1-\boldsymbol{x}_i)\cdot \log(1-\hat{\boldsymbol{x}}_i)\right] \right]. \tag{6.27}$$

Therein, $N_{\boldsymbol{x}}$ is the number of pixels in $\boldsymbol{x}$, $\boldsymbol{x}_i$ is the i-th pixel of $\boldsymbol{x}$, $\hat{\boldsymbol{x}} = A_{\text{dec}}(\boldsymbol{z}, y)$ is the decoded image, and $\boldsymbol{z} = A_{\text{enc}}(\boldsymbol{x}, y)$ is the encoded latent representation of $\boldsymbol{x}$. Note that the pixel values are expected to be normalized in $\boldsymbol{x}_i \in [0, 1]$, and we define $0 \cdot \log(0) = 0$. After training, we freeze the weights and can use the cAE to transform our training data into a low-dimensional latent space. Within the latent space we will now define our cGAN training objective as

$$\begin{aligned} &\min_G \max_D \frac{1}{|\mathcal{S}|} \sum_{(\boldsymbol{x},y)\in\mathcal{S}} \Bigg[\overbrace{D(\boldsymbol{z} \mid y) - D(\tilde{\boldsymbol{z}} \mid y) + \lambda_{\text{gp}} \cdot \ell_{\text{gp}}(\boldsymbol{z})}^{\text{(a)}} \overbrace{-\lambda_{\text{cl}} \cdot \log\left(1 - C(i \mid \tilde{\boldsymbol{x}}, y)\right)}^{\text{(b)}} \Bigg] \\ &\qquad + \lambda_R \mathfrak{L}_R\,, \\ &= \min_G \max_D \mathfrak{L}_{GD} \end{aligned} \tag{6.28}$$

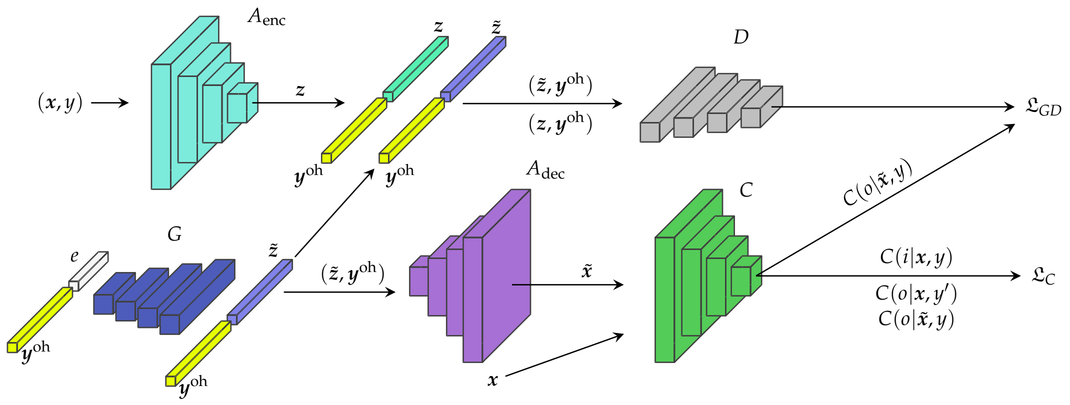

Figure 6.3: Overview of the unified training architecture of cAE, GAN and OvA classifier. Before the GAN and OvA classifier are trained, the cAE is pre-trained on the training data and the frozen weights are used for the rest of the training.

with D and G the critic and generator, $\tilde{z} = G(e, y)$ the latent encoding produced by the generator from a sample $e \sim U(0, 1)$ of a uniform distribution, $\tilde{x} = A_{\text{dec}}(\tilde{z}, y)$ the decoding of the latent code by the generator, $\mathfrak{L}_R$ our low-dimensional regularizer with hyperparameter weight λ_R, and $\ell_{\text{gp}}(z)$ the gradient penalty from Gulrajani et al. [46] with the corresponding hyperparameter λ_{gp}, defined as

$$\begin{aligned} \ell_{\text{gp}}(z) &= \left(\|\nabla_{z'} D(z')\|_2 - 1\right)^2 , \\ z' &= e \cdot z + (1 - e) \cdot \tilde{z} , \\ e &\sim U(0, 1) . \end{aligned} \tag{6.29}$$

We call D a *critic* as in Arjovsky, Chintala, and Bottou [4] and Gulrajani et al. [46] instead of *discriminator* as in the original GAN by Goodfellow et al. [42], because instead of using the JS divergence as a training objective, we use a Wasserstein GAN, for which the value function is defined over the *Kantorovich-Rubinstein duality* [137, pp. 51–]. Wasserstein GANs have some theoretical properties, which makes their optimization more stable. For example, JS GANs suffer from vanishing gradients due to a saturating discriminator, whereas the Wasserstein alternative exhibits linear gradients [4]. The theory of Wasserstein GANs requires the critic to be a 1-Lipschitz function. This constraint was originally enforced by weight clipping [4] but Gulrajani et al. [46] showed, that a gradient penalty behaves more stable during the optimization process. In Eq. (6.28), (a) is the original Wasserstein loss by Gulrajani et al. [46], while (b) is introducing our classification model C into the GAN objective. Intuitively, Eq. (6.28) can be explained as the following:

- The critic D is trained to predict large values for real latent codes ($D(z \mid y)$) and small values for generated ones ($D(\tilde{z} \mid y)$)

- The generator G is trained to minimize $C(i \mid \tilde{x}, y)$ and to maximize $D(\tilde{z} \mid y)$, which forces the generator to produce latent codes $\tilde{z}$, which can not be distinguished from real ones by the critic but for which the reconstructions $\tilde{x}$ are assigned to a low ID probability by the OvA classifier. This interplay between D and C forces the generator to place $\tilde{z}$ at the boundary of the training distribution.

In addition to Eq. (6.28), we need an objective for our classification model C, which is similar to Eq. (6.1) but incorporates the generated boundary samples $\tilde{\boldsymbol{x}}$. This can be defined as

$$\begin{aligned}\min_C \frac{1}{|\mathcal{S}|} \sum_{(\boldsymbol{x},y)\in\mathcal{S}} & \Bigg[-\log(C(i \mid \boldsymbol{x}, y)) - \frac{\lambda_{\text{real}}}{n-1} \left(\sum_{y' \in \mathcal{Y} \setminus \{y\}} \frac{\hat{p}(y)}{\hat{p}(y')} \log(1 - C(i \mid \boldsymbol{x}, y')) \right) \\ & - (1-\lambda_{\text{real}}) \cdot \log(1 - C(i \mid \tilde{\boldsymbol{x}}, y)) \Bigg] \\ = \min_C \mathfrak{L}_C & \end{aligned} \tag{6.30}$$

With the above training objective the classifier for class y learns to predict high in-class values for real inputs $(\boldsymbol{x}, y)$ and low in-class values for other classes $(\boldsymbol{x}, y')$ and generated OoC examples $(\tilde{\boldsymbol{x}}, y)$. The hyperparameter λ_{real} can be used to tune the weight on real and generated examples. During the training phase we alternate between the optimization of G and D (Eq. (6.28)), as well as C (Eq. (6.30)). The complete model is visualized in Fig. 6.3.

Note that one of the assumptions for Lemma 1 (Assumption 1) is that we do not include any generated data into the training objective of the OvA classifiers. We will see in Chapter 8 if this violation has a significant impact on the performance or ability to quantify uncertainties.

6.3 LOW-DIMENSIONAL REGULARIZATION

When leaving out the regularization loss $\mathfrak{L}_R$ in Eq. (6.28), the generator suffers from mode collapse as shown by Ngo et al. [96] and Vernekar et al. [136]. In this case, the points are very close together and sometimes not even on the boundary, which makes them not suitable for class-shielding in all directions. To prevent this from happening, we can penalize the generator to place the points very close to one another. There are many ways to formulate a regularizer which meets the above criteria. Ngo et al. [96] call their regularizer *dispersion loss*, which they define as

$$\begin{aligned} DL &= \frac{|\mathcal{S}|}{\sum_{i=1}^{|\mathcal{S}|} \|G(e_i) - \boldsymbol{\mu}\|_2}, \\ \boldsymbol{\mu} &= \frac{1}{|\mathcal{S}|} \sum_{i=1}^{|\mathcal{S}|} G(e_i), \\ e_i &\sim U(0,1), i = 1, \ldots, |\mathcal{S}|. \end{aligned} \tag{6.31}$$

The generator can minimize the dispersion loss by increasing the average L_2 distance for a set of generated examples to their center of mass. Equation (6.31) is defined on the original learning domain $\mathcal{X}$. It has been shown in other works, that many distance metrics perform poorly in high dimensions [2] and the distance to the nearest neighbor approaches the distance to the farthest neighbor [7] under some very broad assumptions. This is one reason why the effectiveness of this dispersion loss is diminished on high dimensional problems (32×32 RGB images have already 3072 dimensions). Additionally, the value range of the L_2 metric is $[0, \infty]$, and it is generally higher when the number of dimensions increases. Finding a suitable hyperparameter weight to balance the dispersion loss against the other loss components will therefore not be stable across a variety of datasets/dimensions and needs to be retuned on a validation set each time. The usage of the L_2-norm also encourages another undesirable collapse of the generator, where the generated points are pushed to infinity to minimize the dispersion loss.

A good alternative is the *cosine similarity*, which is defined as

$$d_{\cos}(\boldsymbol{z}^{(1)}, \boldsymbol{z}^{(2)}) = \frac{\boldsymbol{z}^{(1)\mathsf{T}} \boldsymbol{z}^{(2)}}{\|\boldsymbol{z}^{(1)}\|_2 \cdot \|\boldsymbol{z}^{(2)}\|_2} = \frac{\sum_{i=1}^{d} z_i^{(1)} \cdot z_i^{(2)}}{\sqrt{\sum_{i=1}^{d} (z_i^{(1)})^2} \cdot \sqrt{\sum_{i=1}^{d} (z_i^{(2)})^2}}, \tag{6.32}$$

for two vectors $\boldsymbol{z}^{(1)}, \boldsymbol{z}^{(2)} \in \mathbb{R}^d$. We can then transform this into an angular distance, which is normalized to $[0, 1]$

$$d_{\mathrm{A}}(\boldsymbol{z}^{(1)}, \boldsymbol{z}^{(2)}) := \frac{1}{\pi} \cdot \underbrace{\arccos\left(\frac{\boldsymbol{z}^{(1)\mathsf{T}} \boldsymbol{z}^{(2)}}{\|\boldsymbol{z}^{(1)}\|_2 \cdot \|\boldsymbol{z}^{(2)}\|_2}\right)}_{\alpha}. \tag{6.33}$$

This metric will still suffer from very high dimensions, as it also utilizes a dot product and the L_2 metric for normalization, but the value range will be bounded to $[0, 1]$, which makes tuning λ_R much easier. Similar to the cAE and cGAN, we will also compute the regularization loss class-wise. As we want to spread the generated samples around our real data, we need to compute the angular distance with respect to an origin inside the class region. For this, let us define $\mathcal{Z}(\boldsymbol{z}, y) := \{\tilde{\boldsymbol{z}} - \boldsymbol{z} \mid \tilde{\boldsymbol{z}} = G(e, y), e \sim U(0,1)\} = \{\bar{\boldsymbol{z}}^{(1)}, \ldots, \bar{\boldsymbol{z}}^{(N_y^z)}\}$ as the set of generated latent codes of class y, which have been normalized to the origin $\boldsymbol{z}$. Our goal is to maximize the average angular distance between all unique pairs of $\mathcal{Z}(\boldsymbol{z}, y)$, which is equivalent to the minimization of

$$\ell_R(\boldsymbol{z}, y) = \frac{2}{N_y^z \cdot (N_y^z - 1)} \cdot \sum_{\substack{\bar{\boldsymbol{z}}^{(i)}, \bar{\boldsymbol{z}}^{(j)} \in \mathcal{Z}(\boldsymbol{z}, y) \\ i<j}} -\log(d_{\mathrm{A}}(\bar{\boldsymbol{z}}^{(i)}, \bar{\boldsymbol{z}}^{(j)})). \tag{6.34}$$

We additionally included the logarithm into this regularizer in order to explicitly target very small angular distances, which often occur in a mode collapse scenario. An illustration of the components in ℓ_R can be found in Fig. 6.4. Finally, averaging over all examples in $\mathcal{S}$, results in the low-dimensional regularization loss in Eq. (6.28)

$$\mathfrak{L}_R = \frac{1}{|\mathcal{S}|} \sum_{y \in \mathcal{Y}} \left[\frac{1}{N_y} \sum_{(\boldsymbol{x}, y) \in \mathcal{S}} \ell_R(A_{\mathrm{enc}}(\boldsymbol{x}, y), y) \right]. \tag{6.35}$$

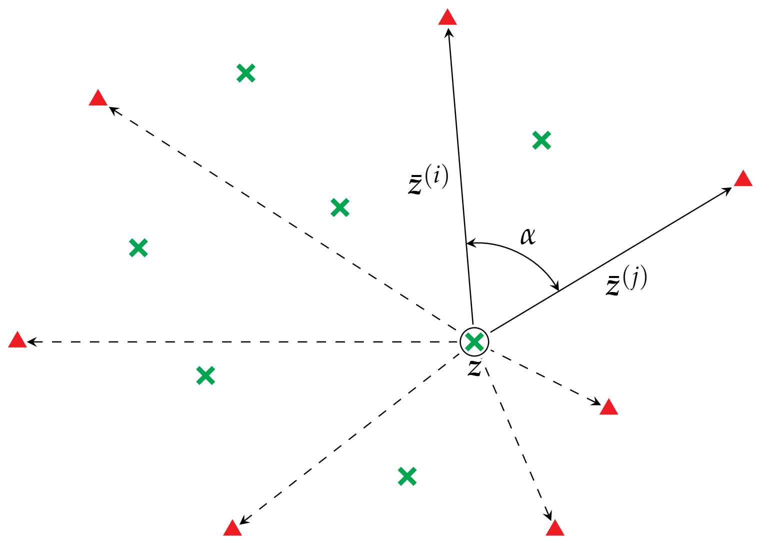

Figure 6.4: Illustration of the low-dimensional regularizer ℓ_R for a single class and a single example. Green crosses correspond to in-class examples and red triangles to OoC examples. The angle α is computed as in Eq. (6.33).

The value range of $\mathfrak{L}_R$ is $[0, 1]$, making it easier to find hyperparameters which are suitable across different datasets, as we will see later.

7 EVALUATING UNCERTAINTY

This chapter will introduce the tasks of OoD and FP detection, which are commonly used as downstream tasks to evaluate the quality of uncertainty estimates. Afterwards, a summary of binary evaluation metrics which are frequently used in related literature are described and explained.

7.1 DOWNSTREAM TASKS

Evaluating the quality of prediction uncertainty is a very difficult task as it is influenced by many factors. Obtaining annotations for prediction uncertainty is virtually impossible as it largely depends on the model and its specific weight configuration. In most cases the quality of uncertainty estimates is measured by utilizing them in downstream tasks like OoD/FP detection, adversarial example detection, semi-supervised or active learning. Another problem, which makes comparing the results between methods difficult, is the lack of a unified evaluation protocol. Recently, there have been efforts to propose specialized benchmark datasets to better understand DNN failure modes [62], OoD detection methods [16, 147] as well as anomaly detection methods [48].

Out-of-Distribution Detection

For evaluating the quality of epistemic uncertainty estimates, the task of OoD detection offers a way of comparing different approaches based on their detection performance. For the task of image classification, the term *Out-of-Distribution* (OoD) means examples that can not be categorized in a meaningful way into the existing class labels or reside in very low density areas of the input space. As an example, consider the task of classifying images of cats and dogs. An image of a car can not be categorized as being a cat or a dog, which is why we consider this to be an OoD input. Similarly, the image of a lion fits semantically into the cat class but when the model has only seen ordinary house cats the lion image is located in a very low density area of the input space. The description "from a low density area" is somewhat blurry. How close can we be to the ID data to still consider it as being OoD? This question is not easy to answer and highlights the difficulty in building dataset benchmarks for this task. It is quite easy to construct an evaluation setup for the task of OoD detection by choosing a training dataset and a number of other datasets which are sufficiently different from it. However, if the chosen OoD datasets are conceptually too different from the ID ones, they can be detected easily and the expressiveness of the results becomes questionable. An effective way to build an evaluation setup with OoD examples which are close to the training distribution, is to split a dataset class-wise. Classes that have been left out during training can serve as particularly difficult OoD cases in the testing phase. This can be combined with other datasets, which are more distinct from the ID data, in order to form a diverse evaluation setup (also used in, e.g., [117, 135]). Outside the training distribution the behavior of a classification model is unpredictable. This is not a specific problem of NNs but rather a principal issue with many types of classification models. Without access to training data for the OoD domain, a model cannot make meaningful predictions. This can also be observed in Fig. 2.2 (b) and (c), where the model outside the training distribution can result in completely different decision boundaries with the same ID performance. Consequently, the decision on this binary classification task is usually made by thresholding the epistemic uncertainty estimates. As the detection accuracy largely depends on the threshold, specialized threshold free evaluation metrics are employed, which will be discussed in Section 7.2.

False Positive Detection

Aleatoric uncertainty quantifies the ambiguity inherent to the ID data (see Fig. 2.2 (a)). Examples which are located in overlapping class regions are very likely to be misclassified. Therefore, aleatoric uncertainty can be used to detect misclassifications (also called False Positives (FPs)). Similar to the case of OoD detection, FPs can be detected by applying a threshold on the aleatoric uncertainty and the same binary evaluation metrics as for the task of OoD detection can be used. Depending on the model and dataset, there is usually a large imbalance in the amount of correct and wrong predictions, which should be taken into account by the evaluation metrics. When reviewing results from this task we should also keep in mind that the larger the classification accuracy, the more difficult it becomes to detect the remaining misclassifications. Comparisons between different models are therefore to be taken with caution.

Adversarial Example Detection

Adversarial examples are perturbed input examples such that the classification model makes a wrong classification. If the image which should be perturbed, is already close to a decision boundary of the classifier, a very small perturbation suffices, and the image remains within the training dataset. However, for decision boundaries which are more distant to the original image (which is more likely for targeted attacks), the perturbations need to be larger. As discussed for the task of OoD detection, model uncertainty is especially high for input space regions with no available data. There, decision boundaries can vary a lot and might be closer to an image than the one within the training data. This is a very likely scenario as DNNs have many parameters which can produce unpredictable decision boundaries. In this case the adversarial examples might leave the subspace covered by the training data in order to keep the overall magnitude of perturbation small. As a reason, there is probably no single uncertainty type that is best for the task of adversarial example detection. We will conduct experiments on this idea in Chapter 8 and try to answer whether uncertainty metrics are at all capable of detecting such attacks. Similar to the two tasks above, we are faced with a binary classification task on a given uncertainty metric and can therefore use the same evaluation metrics. These metrics will be discussed in the upcoming Section 7.2.

7.2 EVALUATION METRICS

7.2.1 *Measuring Accuracy and Calibration*

To measure the influence of UQ methods on the model performance we will measure the standard classification accuracy. In our multi-class setup with K classes, the accuracy of our prediction model $\hat{p}(y \mid \boldsymbol{x})$ can be defined as

$$\text{(Accuracy)} \quad Acc(\mathcal{S}, \hat{p}) = \frac{\sum_{(\boldsymbol{x},y)\in\mathcal{S}} \mathbb{1}_{\{y=\hat{y}^{\text{MAP}}\}}}{|\mathcal{S}|} \tag{7.1}$$

This quantity is often called top-1 accuracy in the literature and is basically the fraction of correctly classified examples.

Another important factor for a classification model is whether the confidence of the model is well calibrated. As shortly described in the beginning of Chapter 4, a model is said to be perfectly-calibrated if its accuracy among all samples with a confidence of $c \in [0, 1]$ is exactly c [26]. The example given there was for a binary classifier, predicting whether it is going to rain. To make the calibration quantifiable in this case, we can compute the calibration error [72, 94]. Formally, for our classification model $\hat{p}(\boldsymbol{x}) \in [0, 1]$ and X, Y random variables for the input and the (binary) class label respectively with a joint distribution of $P(X, Y)$, the calibration error is defined as

$$CE_{\text{p}}(\hat{p}) = \left(\mathbb{E}_P \left[|\hat{p}(X) - \mathbb{E}_P \left[Y \mid \hat{p}(X)\right]|^{\text{p}}\right]\right)^{\frac{1}{\text{p}}} . \tag{7.2}$$

While $\text{p} = 2$ is the most used formula for the calibration error, $\text{p} = 1$ is often referred to as the Expected Calibration Error (ECE) and $\text{p} = \infty$ as the Maximum Calibration Error (MCE) [72]. Computing the CE_{p} on a finite dataset $\mathcal{S}$ requires estimating

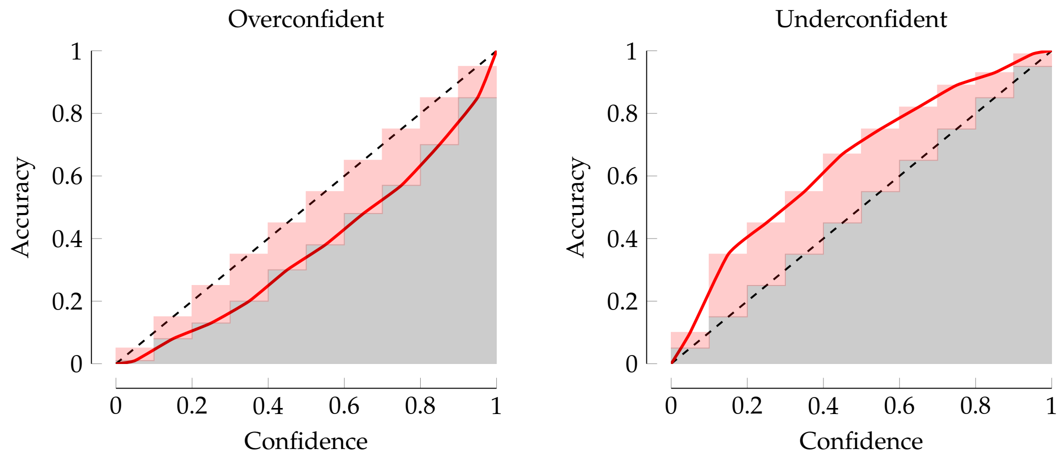

Figure 7.1: Illustration of calibration curves for over- and underconfident predictors. The bars correspond to the bins over which the calibration error is estimated. Grey bars are accuracy values among the bins, while red bars illustrate the calibration error in the bin compared to a perfectly calibrated model (dashed line). Graphic is based on Figure 1 from Guo et al. [47].

the expectations. This is usually done by binning the predictions into a number of N_B equally spaced bins $B_i = \{(\boldsymbol{x}, y) \in \mathcal{S} \mid b_{i-1} < \hat{p}(\boldsymbol{x}) \leq b_i\}, i = 1, \dots, N_B$, with $0 = b_0 < b_1 < \dots < b_{N_B} = 1$ and aggregating them within these bins

$$\begin{aligned}
\bar{\boldsymbol{p}}_i &= \frac{1}{|B_i|} \sum_{(\boldsymbol{x},y)\in B_i} \hat{p}(\boldsymbol{x}), \\
\bar{\boldsymbol{y}}_i &= \frac{1}{|B_i|} \sum_{(\boldsymbol{x},y)\in B_i} y, \qquad\qquad i = 1, \dots, N_B. \\
P(B_i) &= \frac{|B_i|}{|\mathcal{S}|},
\end{aligned} \tag{7.3}$$

Here, $\bar{\boldsymbol{p}}_i$ is the average confidence, $\bar{\boldsymbol{y}}_i$ the frequency of positive labels, and $P(B_i)$ the relative amount of samples in bin i. The calibration error can then be estimated as

$$CE_\mathrm{p}(\hat{p}) \approx \begin{cases} \sum_{i=1}^{N_B} P(B_i) * \|\bar{\boldsymbol{p}}_i - \bar{\boldsymbol{y}}_i\|_\mathrm{p} & \mathrm{p} \neq \infty \\ \max_{i=1,\dots,N_B} |\bar{\boldsymbol{p}}_i - \bar{\boldsymbol{y}}_i| & \mathrm{p} = \infty \end{cases}. \tag{7.4}$$

Figure 7.1 depicts calibration curves of an overconfident (left) as well as an underconfident (right) model. The ECE (Eq. (7.4) with $\mathrm{p} = 1$ and $N_B = 10$) within each bin is visualized with red bars. In a multi-class setup, Y can be defined as the random variable depicting if the prediction of the model is correct or not (e.g., by using the MAP decision rule). Then, $\bar{\boldsymbol{y}}_i$ corresponds to the accuracy of the classifier among all examples in bin B_i.

There are more ways to define calibration in a multi-class setting. We want the winning class probability to be calibrated like in the binary setting but in some settings it is also desirable that the other class probabilities are also calibrated. Kumar, Liang, and Ma [72] give the example of a medical diagnosis system, predicting a patients' tumor to be 50 % benign, 10 % aggressive, and 40 % not a tumor. Different classes can have different levels of risk, e.g., diagnosing a patient with no tumor, although he has an aggressive one, has serious consequences for the life and well-being of the patient. This is why in some cases a multi-class model should be calibrated for every class and not only the winning one. Kumar, Liang, and Ma [72] define the *marginal calibration error* for this purpose. Given a multi-class model $\hat{p}(y \mid \boldsymbol{x})$ and X and Y being similar random variables as above, for the input and class label and with a joint distribution of $P(X, Y)$, the marginal calibration error is defined as

$$MCE_{\mathrm{p}}(\hat{p}) = \left(\sum_{k=1}^{K} \boldsymbol{w}_k \mathbb{E}_P \left[(\hat{p}(y=k \mid X) - P(Y=k \mid \hat{p}(y=k \mid X)))^{\mathrm{p}} \right] \right)^{\frac{1}{\mathrm{p}}} . \quad (7.5)$$

Here, $\boldsymbol{w}_k \in [0,1]$ is a weight for each class, which makes it possible to increase the importance of one class in comparison to the others. For equally important classes we just set $\boldsymbol{w}_k = \frac{1}{K}$. Given a finite dataset $\mathcal{S}$, MCE_{p} is estimated similarly as in Eq. (7.4) and summed over all classes.

The most frequently used evaluation metric throughout the UQ literature is the ECE, which we will also utilize in our experimental evaluation.

7.2.2 *Evaluating Binary Classifiers*

Using epistemic and aleatoric uncertainty estimates to tackle the problem of OoD and FP detection is a common way to measure the uncertainty quality. By applying a threshold on the uncertainty we are essentially solving a binary classification problem. For the task of OoD detection we use a dataset $\mathcal{S}_{\boldsymbol{x}} = \mathcal{S}_{\boldsymbol{x},\mathrm{in}} \cup \mathcal{S}_{\boldsymbol{x},\mathrm{out}}$ consisting of ID and OoD data. Given a threshold τ as well as a function which quantifies epistemic prediction uncertainty $u(\boldsymbol{x})$ (we use a generic placeholder because uncertainties can be computed in many ways), we can define standard binary classification metrics

$$\text{(True Positives)} \qquad TP = |\{\boldsymbol{x} \in \mathcal{S}_{\boldsymbol{x},\mathrm{in}} \mid u(\boldsymbol{x}) \leq \tau\}| \qquad (7.6)$$

$$\text{(True Negatives)} \qquad TN = |\{\boldsymbol{x} \in \mathcal{S}_{\boldsymbol{x},\mathrm{out}} \mid u(\boldsymbol{x}) > \tau\}| \qquad (7.7)$$

$$\text{(False Positives)} \qquad FP = |\{\boldsymbol{x} \in \mathcal{S}_{\boldsymbol{x},\mathrm{out}} \mid u(\boldsymbol{x}) \leq \tau\}| \qquad (7.8)$$

$$\text{(False Negatives)} \qquad FN = |\{\boldsymbol{x} \in \mathcal{S}_{\boldsymbol{x},\mathrm{in}} \mid u(\boldsymbol{x}) > \tau\}| \qquad (7.9)$$

Note that in the above definition the ID class took the part of the positive class. Based on these quantities we can compute metrics like

$$\text{(Precision)} \qquad Prec(\mathcal{S}, u) = \frac{TP}{TP + FP}, \qquad (7.10)$$

$$\text{(Recall)} \qquad Rec(\mathcal{S}, u) = \frac{TP}{TP + FN}, \qquad (7.11)$$

$$\text{(False Positive Rate)} \qquad FPR(\mathcal{S}, u) = \frac{FP}{FP + TN}. \qquad (7.12)$$

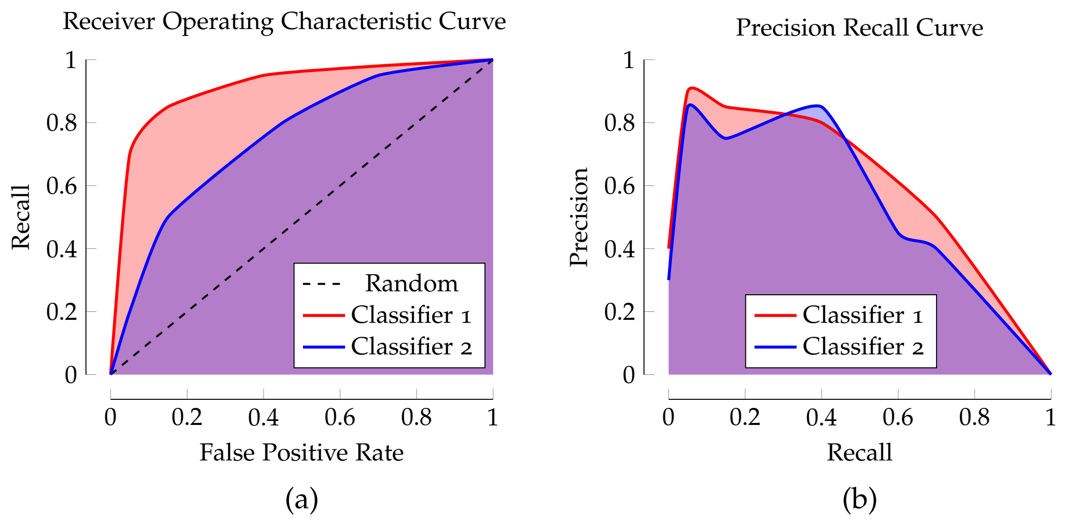

Figure 7.2: Receiver Operating Characteristic (ROC) Curve (a) and Precision Recall (PR) Curve (b) for two binary classification models. The dashed line in (a) is representing a random classifier.

Recall is also called *sensitivity* while $1 - FPR$ is often called *specificity*. The problem with these metrics is that they heavily depend on the choice of the threshold τ. If we vary the threshold $\tau \in \{u(\boldsymbol{x}) \mid \boldsymbol{x} \in \mathcal{S}\}$ and compute the above metrics for each choice, we can get a better intuition on the performance of the binary classifier. The *Receiver Operating Characteristic* (ROC) *Curve* does exactly that and plots the False Positive Rate (FPR) against the recall for varying threshold levels. By increasing the threshold, we can trade a higher recall against a higher FPR. In Fig. 7.2 (a) we can see that "Classifier 1" has clearly a better ROC curve than "Classifier 2", as it has a consistently higher recall at every level of FPR. Another plot that reveals more performance insights of a binary classifier is the *Precision Recall* (PR) *Curve*, which plots the recall against the precision.

Figure 7.2 (b) shows an exemplary plot of such a precision recall curve. In the given example we can not clearly say which classification model is performing better. A qualitative review of ROC and PR curves is always good practice, but we need a way to quantitatively summarize them in order to better compare different classification models. To achieve this we can compute the areas under the respective curves. We will call them the Area under Receiver Operating Characteristic (AUROC) and Area under Precision Recall (AUPR). As both curves are confined in a unit square, the area will always be in the interval $[0, 1]$. Theoretically a classification model can not have an AUROC lower than 0.5 (random guessing). If a classifier is indeed consistently worse than random guessing, therefore having a ROC below the dashed line in Fig. 7.2 (a), we can invert the decision function and end up with an area under the ROC curve of greater than 0.5. However, for the application to FP, OoD and adversarial example detection based on UQ metrics, it is possible to achieve an AUROC of lower than 0.5. In this case we have a special meaning attached to the interpretation of uncertainties. Inverting the decision function and defining ID examples to have a high uncertainty makes no sense semantically. In case of the PR

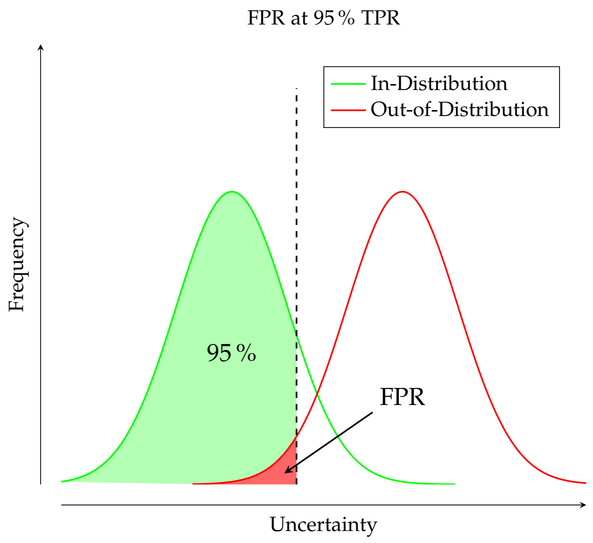

Figure 7.3: Illustration of the binary classification metric *FPR@95%TPR*. Green and red curve are frequencies of in- and out-of-distribution examples, respectively. Abscissa and ordinate are depicting uncertainty and frequency, respectively. Dashed line is representing the applied threshold at which at least 95 % of ID samples are below.

curve, the lower bound of the area depends on the amount of positive and negative samples [113]. Taking the ID samples as our positive class, the lower bound can be computed as

$$\text{AUPR} \geq \frac{|\mathcal{S}_{\boldsymbol{x},\text{in}}|}{|\mathcal{S}|}. \tag{7.13}$$

Davis and Goadrich [25] show that in case of highly imbalanced datasets, the PR curve is a more suitable metric. Because of the sensitivity to class imbalance, it is advisable to compute the AUPR twice, where each of the two classes serves as the positive class once [52]. The AUROC and AUPR are beneficial if the goal is to summarize an average model performance into a single scalar value. However, it is far away from a practical application, where one has to settle for a single threshold in order to perform the actual binary classification. From this point of view, we either need auxiliary OoD datasets to determine a suitable value for the threshold, or choose it solely on the ID data. The former would require additional auxiliary datasets, which can not be assumed to be readily available. In the latter case, a suitable choice is to set the threshold in such a way that a given quantile of ID data points is classified as true positive

$$\begin{aligned} \min_{\boldsymbol{x}\in\mathcal{S}_{\boldsymbol{x},\text{in}}} \quad & u(\boldsymbol{x}) \\ \text{s.t.} \quad & \frac{|\{\boldsymbol{x}\in\mathcal{S}_{\boldsymbol{x},\text{in}} \mid u(\boldsymbol{x}) \leq \tau\}|}{|\mathcal{S}_{\boldsymbol{x},\text{in}}|} \cdot 100 \geq q \cdot \\ & q \in (0, 100) \end{aligned} \tag{7.14}$$

Equation (7.14) can be solved by performing a line-search along all ID examples sorted with respect to their predicted uncertainty. The resulting threshold τ^q can then be used in practical applications. A measure of quality for τ^q is then the FPR on the OoD datasets we evaluate our uncertainty metric on. This metric is called *FPR@q%TPR* in related literature (e.g., [83]), where usually $q = 95$. Figure 7.3 illustrates this in an idealized example, where the fraction of OoD examples classified as being ID results in the FPR@95%TPR. In the following we will also call this metric FPR95 for brevity.

8 EXPERIMENTS

The results of a thorough qualitative and quantitative evaluation will be discussed in this chapter. A collection of related methods and the proposed approaches from Chapters 5 and 6 will be benchmarked on 4 tasks and a variety of datasets. Each section concludes with the findings of the respective quantitative results.

8.1 DATASETS

This section introduces the datasets which will be used for qualitative and quantitative analysis of the proposed methods discussed in Chapters 5 and 6. The datasets are sorted chronologically and in increasing order of complexity. A summary of all datasets and their characteristics can also be found in Table 8.1.

MNIST

The Modified NIST (MNIST) dataset was published by Lecun et al. [77] in 1998. It consists of handwritten digits (0-9) from a total of 500 different writers which were normalized in size to fit a 20×20 pixel area in the center of a 28×28 image (Fig. 8.1 (a)). The original images were binary but due to the interpolation during the normalization, intermediate gray levels were introduced. Due to the classification accuracy surpassing 99.5 % with modern DNNs, the classification problem on the MNIST dataset can be considered to be solved. Nevertheless, the dataset is still often used for studying new methods and theoretical results, as the small size and exact problem definition are ideal for qualitative analyses. In total the dataset consists of 60 000 training and 10 000 testing images.

EMNIST-Letters

The Extended MNIST (EMNIST) dataset by Cohen et al. [19] utilizes the parts of the NIST database that were not used by Lecun et al. [77] and contains samples of handwritten digits and letters. In the EMNIST-Letters subset, only the letters were used and preprocessed in the same manner as in the MNIST dataset (a-z, A-Z,

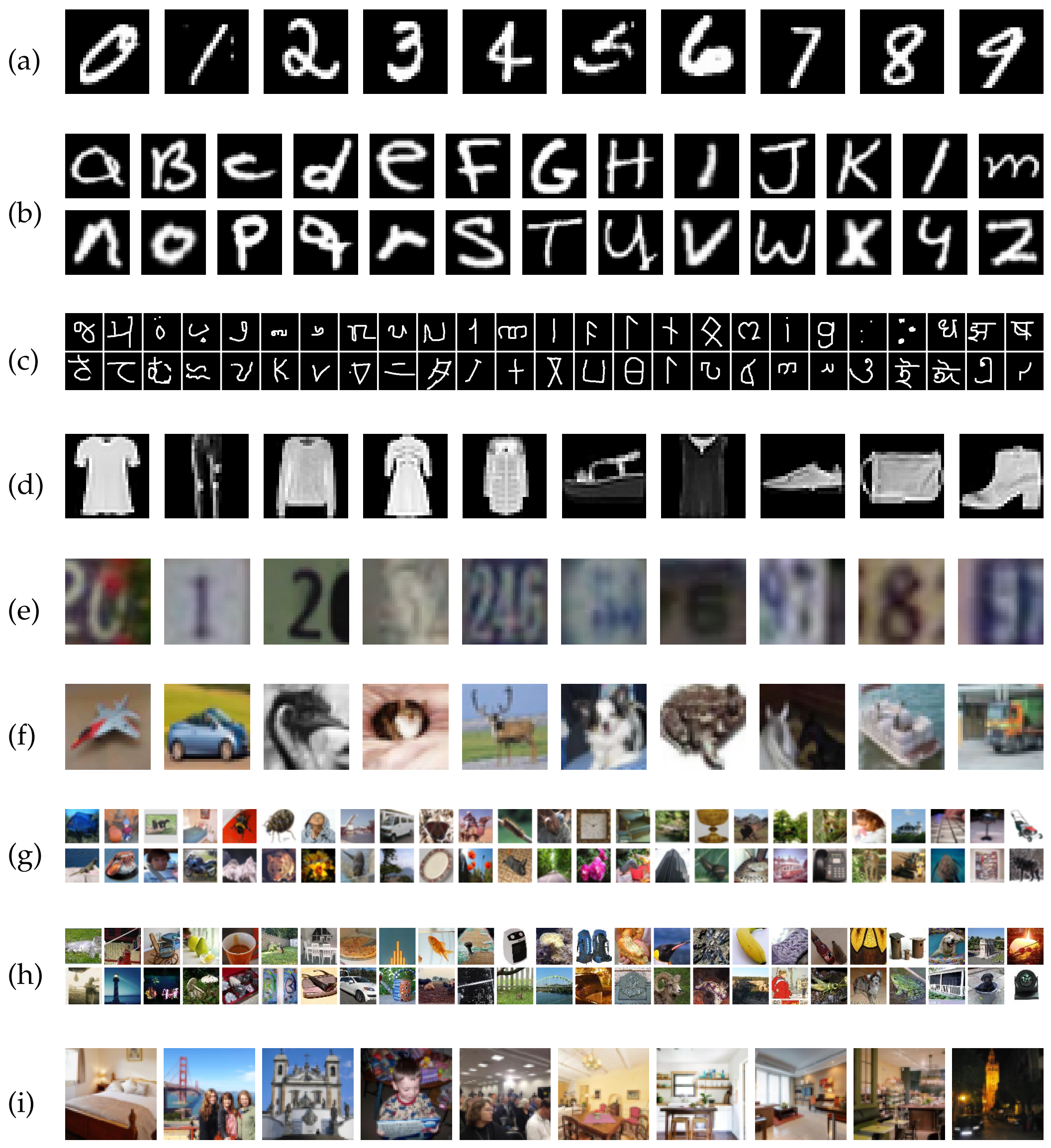

Figure 8.1: Examples of all used datasets. MNIST (a) [77], Fashion-MNIST (b) [141], EMNIST-Letters (c) [19], Omniglot (d) [74], SVHN (e) [95], CIFAR10 (f) [68], CIFAR100 (g) [68], Tiny ImageNet (h) [76] and LSUN (i) [143]. Note that for the Omniglot, CIFAR100 as well as Tiny ImageNet datasets only a subset of the available classes $(1 - 50)$ are displayed.

Fig. 8.1 (b)). For training, there are 124 800 samples and for testing 20 800. When using the MNIST dataset as our ID set, the EMNIST-Letters are a good candidate for an OoD set, as the methodology of capturing and preprocessing is the same and letters are semantically close to digits.

Omniglot

The Omniglot dataset by Lake, Salakhutdinov, and Tenenbaum [74] contains stroke data for 1623 characters out of 50 alphabets and written by 20 different people (Fig. 8.1 (c)). Lake, Salakhutdinov, and Tenenbaum [74] split the dataset into a training set of 30 alphabets and an evaluation set of 20 alphabets containing 19 280 and 13 180 samples, respectively. For our purpose only the evaluation set is used, and the pixel values are inverted to match the "white on black" samples of the MNIST dataset. Similar to the EMNIST-Letters dataset, Omniglot samples are semantically similar to handwritten digits but still distinguishable from them, making them suitable for evaluation as an OoD dataset.

Fashion-MNIST

The Fashion-MNIST dataset by Xiao, Rasul, and Vollgraf [141] was collected by scraping 70 000 fashion product pictures from the Zalando online shop. After scraping, the images which were originally in RGB format, where transformed into 28×28 gray-scale images similar to the MNIST dataset (Fig. 8.1 (d)). The training and testing sets contain 60 000 and 10 000 samples, respectively. Again, we will only utilize the test set for evaluation purposes. In contrast to the EMNIST-Letters and Omniglot datasets, Fashion-MNIST samples are clearly different from MNIST samples but are still within the same gray-scale image space. Even though these are easy to detect as OoD for humans, some UQ methods might struggle with this, which is why we include them as a representative for the OoD domain.

Street View House Numbers (SVHN)

The Street View House Numbers (SVHN) dataset by Netzer et al. [95] consists of house numbers collected from over 600 000 Google Street View images. Amazon Mechanical Turk (AMT) together with a sliding window house-numbers detector was used to extract and label the house numbers. In total, the dataset contains 630 420 examples, which are split across 73 257 training and 26 032 testing. The remaining 531 131 examples constitute an additional subset which are easier to classify. Besides the full image of the house numbers there is also a cropped format, centering the individual digits of each house number in a 32 × 32 MNIST-like format (Fig. 8.1 (e)). To be compatible with the MNIST dataset, we use the testing set with the cropped format and transform the images from RGB to gray-scale. Samples from the SVHN dataset are semantically similar to handwritten digits from the MNIST dataset but are extracted from natural scenes, making the SVHN set an interesting candidate for the OoD domain w.r.t. MNIST.

*CIFAR*10 / *CIFAR*100

The CIFAR10 and CIFAR100 datasets by Krizhevsky [68] are based on the 80 million tiny images dataset [132] which was crawled from the web based on the WordNet database of English nouns [91]. CIFAR10 and CIFAR100 differ in the number of classes, where CIFAR10 contains 10 classes (e.g., *airplane*, *automobile* and *bird*) and CIFAR100 contains 100 classes (e.g., *apple*, *aquarium fish* and *baby*). For the CIFAR10 dataset each of the 10 classes has 6000 annotated samples, while the CIFAR100 dataset only has 600 samples per class. This amounts to 60 000 samples for each of the two datasets, where the classes of the CIFAR100 dataset are disjoint from the ones in the CIFAR10 dataset. In both variants the 60 000 samples are split into 50 000 for training and 10 000 for testing. All Images have a size of 32 × 32 and 3 color channels (RGB) (Fig. 8.1 (f)(g)).

Tiny ImageNet

The Tiny ImageNet dataset by Le and Yang [76] is a subset of the large ImageNet database [27] and was originally published in conjunction to a visual recognition challenge in 2015 [76] (Fig. 8.1 (h)). There are a total of 120 000 images, split across training, evaluation and testing sets with 100 000, 10 000 and 10 000 examples each, respectively. For training, there are 200 annotated object classes, with annotation for the training and evaluation sets. Because of the fact, that the annotations for the testing set are not publicly available, we will use the official evaluation set as our test set. All images have a size of 64 × 64 with RGB color channels and are thus 4 times larger in dimension than the datasets previously described.

LSUN

The LSUN dataset [143] is a large collection of 10 scenes and 20 object categories containing a total of approximately 69 million images. Scene categories refer to a complete scene rather than a single object (e.g., *bedroom*, *bridge* and *kitchen*) while object categories refer to single items (e.g., *bike*, *car* and *plant*). Yu et al. [143] use an

Dataset	Year	Image Format	Training	Evaluation	Test
MNIST [77]	1998	28×28, grayscale	**48 000**	**12 000**	**10 000**
EMNIST-Letters [19]	2017	28×28, grayscale	99 200	**24 800**	**20 800**
Omniglot [74]	2015	105×105, binary	15 424	**3856**	**13 180**
Fashion-MNIST [141]	2017	28×28, grayscale	48 000	**12 000**	**10 000**
SVHN [95]	2011	32×32, RGB	58 606	**14 651**	**26 032**
CIFAR10 [68]	2009	32×32, RGB	**40 000**	**10 000**	**10 000**
CIFAR100 [68]	2009	32×32, RGB	**40 000**	**10 000**	**10 000**
Tiny ImageNet [76]	2015	64×64, RGB	**80 000**	**20 000**	**10 000**
LSUN [143]	2015	$\geq 256 \times 256$, RGB	7 916 299	**1 979 074**	**3000**

Table 8.1: Summary of training, evaluation and testing datasets and their characteristics. Subsets which are used in the experiments are marked in bold.

ID	Out-of-Distribution
MNIST 0-4	MNIST 5-9, CIFAR10, EMNIST-Letters, Omniglot, Fashion-MNIST, SVHN
CIFAR10 0-4	CIFAR10 5-9, LSUN, SVHN, Fashion-MNIST, MNIST
CIFAR100 0-49	CIFAR100 50-99, LSUN, SVHN, Fashion-MNIST, MNIST
Tiny ImageNet 0-99	Tiny ImageNet 100-199, SVHN, Fashion-MNIST, MNIST

Table 8.2: ID and OoD datasets used for the quantitative evaluation of OoD and FP detection.

active learning framework within Amazon Mechanical Turk (AMT) to efficiently annotate such a large dataset. The images have different sizes in RGB format, which will be converted to an appropriate size depending on the ID dataset. For our purposes we will only utilize the evaluation set of scene samples, which contains a total of 3000 images with 300 per class (Fig. 8.1 (i)).

8.2 TRAINING AND EVALUATION PROTOCOLS

In the experiments we will evaluate the proposed gradient metrics, UQGAN and UQGAN-MCD against numerous established baselines (see Section 8.5.1) on the four tasks *Classification Accuracy and Calibration*, *False Positive Detection*, *Out-of-Distribution Detection* and *Adversarial Example Detection* (see Section 7.1). From all the datasets mentioned in Section 8.1 we will build a number of diverse evaluation protocols for the task of OoD detection. These datasets will then also provide the evaluation setting for the other three tasks. As briefly mentioned in Chapter 7, we will follow other works [117, 135] and split our ID datasets class-wise in two equally sized and non-overlapping subsets. The first subset will be used as an ID training set, while the other half will be utilized as particularly difficult OoD cases. For the training of our classification models, we will utilize the MNIST, CIFAR10, CIFAR100 and Tiny ImageNet datasets. All these datasets do not have a dedicated evaluation set, which is why we additionally split the training data randomly into 80 % for training and 20 % for evaluation. The evaluation sets will then be used for hyperparameter exploration and model selection based on the highest classification accuracy. Depending on the ID dataset, we will assemble a number of OoD datasets for the task of OoD detection. A summary of the respective ID and OoD datasets can be seen in Table 8.2.

We will use different architectures to adapt to the complexity of the data and not to risk overfitting. For MNIST 0-4 the LeNet model [77] (see Section 3.6.1) is a good fit in terms of capacity, while for the CIFAR10, CIFAR100 and Tiny ImageNet datasets we will train a ResNet 18 [50] (see Section 3.6.2). Choosing a single ResNet architecture for all the last three mentioned datasets allows better comparison between the performance and uncertainty estimates. There are definitely models which will achieve higher classification accuracy on any of the chosen ID datasets but as this work is more about UQ rather than maximizing classification performance, we will stick to these commonly used architectures. However, we still need to make sure that we achieve comparable classification performance with respect to a common baseline in order to detect any detrimental effects of UQ methods on the model performance. Our general baseline will be the respective architecture trained with the standard cross-entropy loss for multi-class problems, while we use its Maximum Class Probability (MCP) as the uncertainty measure. This is in line with the work by Hendrycks and Gimpel [52]. The UQGAN will also be build with different combinations of cAE models. While using a very small convolutional architecture on the MNIST dataset, a small cAE based on the ResNet model is built for the CIFAR10, CIFAR100 and Tiny ImageNet datasets. An advantage of the UQGAN architecture is that the GAN is learned within the low-dimensional latent space of the cAE. This makes it possible to keep the size of the GAN architecture very small. Both, the generator and critic will be an MLP, with the generator consisting of 3 hidden layers of 1024, 512 and 256 neurons each. The critic is also built with 3 hidden layers but containing 512 neurons each.

The task of *Classification Accuracy and Calibration* will be evaluated using the standard accuracy performance metric, while the calibration is measured using the ECE with 15 bins. Additionally, we will also have a look at calibration curves to infer if a model is over- or underconfident.

False Positive Detection is a binary classification task based on the aleatoric uncertainty and as such will be evaluated with the AUROC, which we then call AUROC S/F (success/failure).

Out-of-Distribution Detection is also a binary classification task and additionally to the AUROC, we will report results with the AUPR-In and FPR95 evaluation metrics. The addition "-In" means that we are considering the ID data as the positive class. As discussed in Chapter 7, the lower bound of the AUPR is the fraction of positive examples of the complete dataset (including ID and OoD data). In our evaluation setup the number of OoD examples is much higher than the ID examples. To have as much expressiveness (or value range) of the performance measure as possible, we need to choose the ID data as the positive class. The FPR95 evaluation metric is important for practical applications as it indicates the false positive error rate when guaranteeing a fixed true positive rate of 95 %, which is set based on the ID data. OoD inputs are increasing the distribution uncertainty, which is part of the total epistemic uncertainty. Therefore, we will use the epistemic prediction uncertainty from models separating into different types of uncertainty. Otherwise, if the model does not explicitly differentiate between uncertainty types, we take the defined uncertainty/confidence measure.

Finally, *Adversarial Example Detection* is just another binary classification task which we will evaluate with the AUROC and FPR95 evaluation metrics for different levels of perturbation strength. We will additionally report the success rate of the attacks (percentage of adversarial examples that result in a different predicted class than the original image) for some experiments. To keep the computational effort manageable, we will sample a random subset of 1000 images of each ID test set and generate another 1000 adversarial examples such that we have a balanced dataset for testing. The ID examples will be the positive class in this case. For generating the adversarial examples we will use the FGSM [43], PGD [88] and PGD applied to an objective targeting the least likely class, as proposed by Kurakin, Goodfellow, and Bengio [73]. All three approaches will be used in a white-box setting, where the methods have full access to the network which is attacked.

For all results that are summarized in a single performance measure, experiments will be repeated 5 times with different seeds for the random number generator (and therefore different weight initialization). The standard deviation will be reported alongside, indicating the robustness of the evaluated methods. All approaches are implemented and evaluated in Python using PyTorch [103] to model the NNs.

8.3 summary of hyperparameter studies

8.3.1 *Metric and Layer Choice for Gradient Metrics*

In Chapter 5 we discussed several ways of summarizing the high dimensional gradient vector over all model parameters. It remains open which of these metrics to summarize gradients works best when measured on the tasks of FP and OoD detection. Additionally, we discussed that it can be beneficial in terms of computational cost to compute the gradient metrics only for a subset of layers. To take advantage of the backpropagation algorithm, the cost can be reduced the most if gradients are computed for layers as close to the output as possible. Appendix B.1 contains a parameter study for the choice of gradient metric and the layers the gradients are computed over. Therein, Fig. B.2 summarizes the results for the different gradient metrics on all ID evaluation datasets. Applied to the cross-entropy loss with MAP target (see Chapter 5), we observe for the L_p-norms that higher values of p (i.e., more similar to the supremum norm) tend to perform better in terms of AUROC, FPR95 and AUPR-In on the task of OoD detection. This is in contrast to the results of Huang, Geng, and Li [60], who compute L_p metrics on a KLD loss to a uniform class prediction (see Section 5.2). However, the KLD between the prediction and a uniform class distribution needs to be interpreted opposite to the cross-entropy over a MAP target. A high gradient on this loss is expected to be found on ID examples, while the gradient on the cross-entropy should have a low magnitude on ID inputs. This difference in the loss formulation is also explaining the contradicting results. Except for the MNIST evaluation setting, higher values of p tend to perform better in terms of AUROC, FPR95 and AUPR-In. On the Tiny ImageNet 0-99 dataset, the absolute difference between L_1 and L_∞ is 14.21 %, 8.12 %, 5.09 % for AUROC, FPR95 and AUPR-In, respectively. Similar trends can be observed on the other datasets, however the results with regard to the AUPR-In evaluation metric are not always consistent to this result. In general the max and min metrics are performing best, while the average gradient is a very poor statistic on the analyzed tasks. Between max and min, the minimum gradient entry is performing slightly better, especially on the higher dimensional Tiny ImageNet 0-99 dataset, where it achieves a 6.2 % relative improvement in AUROC over the max metric. The fact that maximum and minimum are overall better statistics than L_∞ suggests that the direction of the gradient plays a role in distinguishing unknown inputs from ID examples.

In terms of the layer choice, Fig. B.1 summarizes an ablation study over the Tiny ImageNet 0-99 evaluation set and a ResNet 18. There, two settings were evaluated. In the first setting, each layer of the ResNet 18 is evaluated individually. In the second setting, multiple layers are considered in a cumulative fashion, starting from the output of the network. Therefore, *Layer* 5 corresponds to the last layer, *Layer* 4-5 corresponds to the two last layers and so on. The second setting was chosen in this cumulative way in order to gain maximum benefits from the computational cost reduction of the backpropagation algorithm. On both settings the differences are marginal, and the results suggest that considering only layers close to the output is sufficient for this approach. When analyzing the FPR95 results, layers which are close to the output do indeed achieve slightly better results, as contemplated in Chapter 5. However, for both the AUROC and FPR95 evaluation metrics, results have an absolute difference of 0.38 % and 1.13 %, respectively. As the standard deviation is comparatively high, these results cannot be considered significant. We can conclude that the choice of layers on which we compute gradient metrics is largely irrelevant for the detection performance of OoD inputs. This result is also in line with the observations of Huang, Geng, and Li [60].

Going onwards, we will choose the minimum metric and compute the gradient over all model parameters whenever numerical results are given.

8.3.2 *UQGAN Hyperparameters*

The UQGAN method as described in Chapter 6 has 4 hyperparameters which are inherent to the methodology (λ_R, λ_{cl}, λ_{real} and the cAE latent dimension). Appendix B.2 contains results of a study over all 4 hyperparameters on MNIST 0-4 and CIFAR10 0-4. For λ_{cl}, which controls the impact of the classification model on the generator (see Eq. (6.28)), we can see that for $\lambda_{\text{cl}} \in [1, 4]$ we achieve the best results. Noticeably, larger values impact the AUPR-In and FPR95 negatively. Although $\lambda_{\text{cl}} = 0$ has a similar effect on the two performance measures, it is not as notable as for high values. This is due to the low-dimensional regularizer forcing the generator to spread the generated samples apart with respect to the class centers. If $\lambda_R = 0 = \lambda_{\text{cl}}$, then Eq. (6.28) would recover the standard Wasserstein loss for GANs and the generator would learn to replicate the training distribution, which would destroy the classifier training objective (Eq. (6.1)). Analyzing the results for λ_{real} (Fig. B.4), we can see that for $\lambda_{\text{real}} \in \{0, 1\}$ there is a clear loss in all measured evaluation metrics. On the one hand, this means that 100 % generated data is detrimental for the classification model but on the other hand, 100 % real data is nearly equally detrimental. This indicates that the generated data has a significant positive effect on the classification accuracy, FP and OoD detection. For $\lambda_{\text{real}} \in [0.1, 0.9]$ the results are quite stable. Overall the best results are achieved with $\lambda_{\text{real}} \in [0.5, 0.6]$. In case of λ_R, the influence of different values is mostly apparent on the low-dimensional MNIST 0-4 dataset. There, larger values of λ_R are increasing the results continuously, up until a regularizer weight of 32. On CIFAR10 0-4 this effect stops at $\lambda_R = 4$, with worse performance for larger weights on the regularizer. Combined we can see that the low-dimensional regularizer is indeed improving the class shielding by the generated OoC examples. The influence of the last hyperparameter, the latent dimension of the cAE, is visualized in Fig. B.6. For both datasets we can see clear local maxima/minima on the performance metrics. On MNIST 0-4 this is for $d = 32$ and for CIFAR10 0-4 it is $d = 128$. Both extremes, increasing and decreasing the number of dimensions, reduce the performance. Although the accuracy is also influenced, it is to note that mostly evaluation metrics measuring the OoD detection performance vary. In general, we can say that the latent dimension should be chosen to allow the cAE to reconstruct the inputs with sufficient visual quality. A higher number of dimensions has a negative effect on the effectiveness of the regularizer and the class-shielding.

As a summary we can say that the hyperparameters of the UQGAN approach are mostly stable and that reasonable hyperparameter choices yield good results.

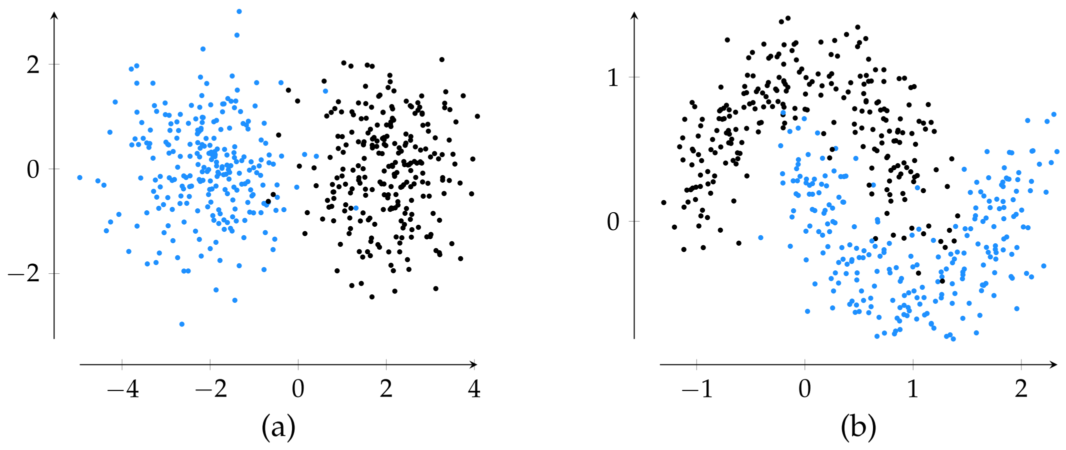

Figure 8.2: Two toy datasets within a two-dimensional space. (a) two Gaussians and (b) two interleaving half circles (also called "two moons" [14]).

8.4 QUALITATIVE RESULTS

8.4.1 *Toy Datasets*

Two-dimensional toy datasets can give valuable insights into methods and highlight certain strengths and pitfalls, as uncertainties can be visualized easily. Figure 8.2 shows simple toy datasets of two Gaussian distributions (Fig. 8.2 (a)) and two interleaving half-circles (Fig. 8.2 (b)), which is also called "two-moons" in the literature [14]. The two Gaussian distributions offer the most basic classification problem and can be constructed so that it is solvable by linear classifiers. On the other hand, the two-moons dataset is more complicated as it requires a non-linear decision boundary to solve the classification problem. We will analyze the gradient metrics as well as the UQGAN on these two datasets and compare the resulting uncertainties to the MCP baseline by Hendrycks and Gimpel [52]. For this we will train a small MLP with 3 hidden layers of size 512 each and LeakyReLU activation function with a negative side slope of 0.01 on each of the toy datasets. The UQGAN will be trained via Eqs. (6.1) and (6.28) which we derived in Sections 6.1 and 6.2 and the MCP baseline will use the standard cross-entropy loss. Gradient metrics are then computed on the same network as for the MCP baseline with the min aggregation metric and on the cross-entropy with MAP target.

Figure 8.3 shows an overview of the resulting uncertainties. As can be seen in the first row, the MCP baseline follows the decision boundary with homogeneously low confidence along the path. In contrast to that, the minimum gradient is producing smaller uncertainties where the two classes overlap. This is mostly apparent in the Gaussian example but can also be observed to a certain degree on the two-moons dataset. As discussed in Chapter 5, gradient metrics quantify model uncertainty, which is part of the epistemic uncertainty. The examples in Fig. 8.3 show that indeed, gradient metrics are able to quantify epistemic uncertainty without including too

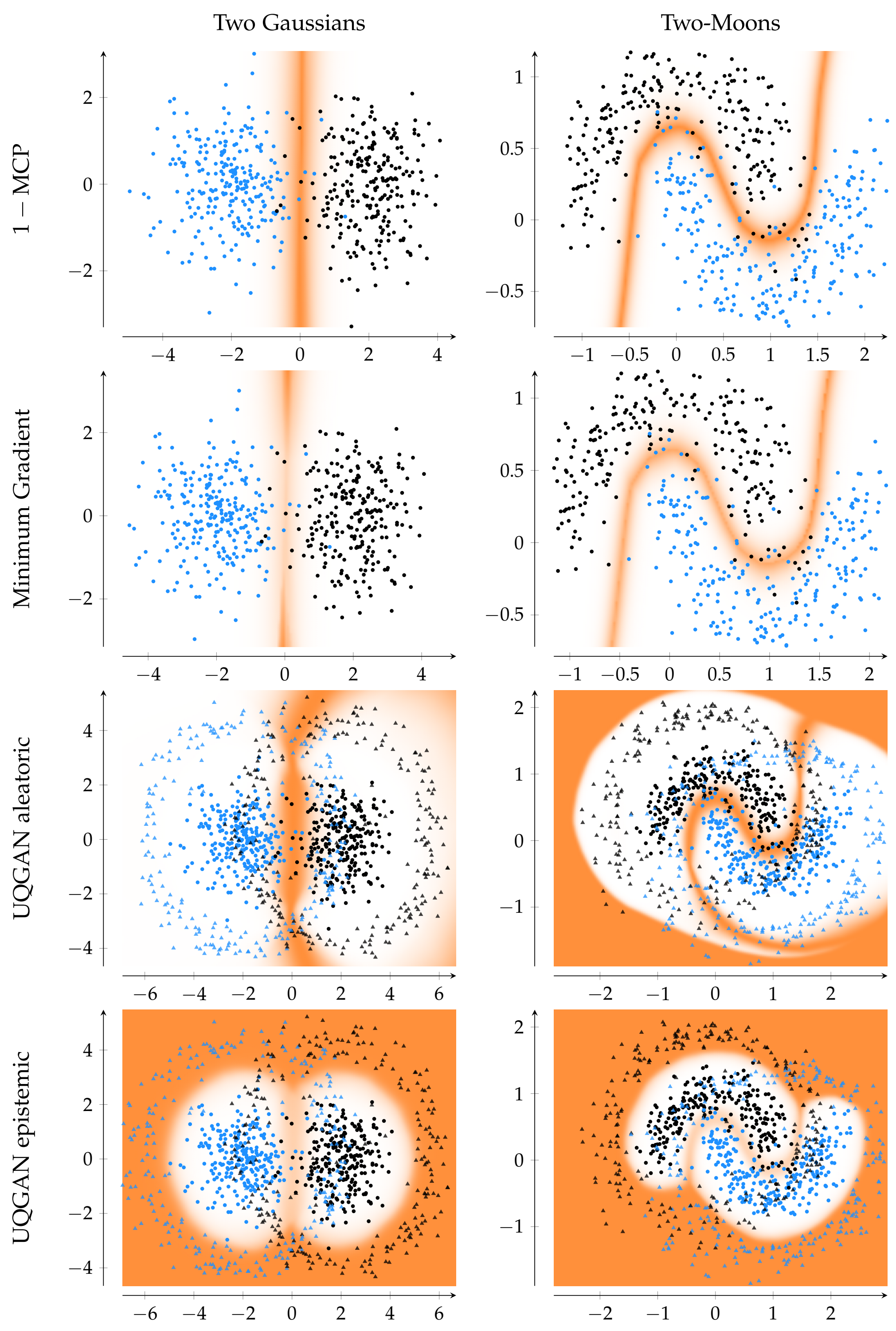

Figure 8.3: Uncertainty heatmaps for the inverse maximum class probability (first row, Section 4.1.1), minimum gradient (second row, Eq. (5.3)), UQGAN aleatoric uncertainty (third row, Eq. (6.20)) and UQGAN epistemic uncertainty (last row, Eq. (6.22)). You can see the results on the Gaussian (left column) and two-moons (right column) toy datasets. White corresponds to low and orange to high uncertainty. In the UQGAN plots you can also see the generated OoC data with triangular marks and the respective class color.

much of the aleatoric part. However, both, MCP and gradient metrics, predict low uncertainties far away from the decision boundary, making them unsuitable to detect OoD inputs. Focusing on the aleatoric uncertainty predicted by the UQGAN approach (Fig. 8.3 third row), we can see that in both cases the region of the class overlaps produces high aleatoric uncertainty. It is also very noticeable that far away from the ID data the model predicts high aleatoric uncertainty. This is a result of the ϵ term added in the computation of $\hat{p}(y \mid \boldsymbol{x})$, as discussed at the end of Section 6.1.1. We can however mask out this region by considering the epistemic uncertainty, which is visualized in the last row of Fig. 8.3. There we can also observe that the aleatoric and epistemic uncertainty are mostly complementary, predicting a low epistemic uncertainty in the overlapping class regions. In contrast to the MCP and minimum gradient, UQGAN is able to predict high epistemic uncertainty for the complete OoD regime. This is only possible because of the generated OoC examples, which are visualized as triangles and in the same color as the respective class. Covering the whole boundary and being still close to the ID data, the OoC examples enable the OvA classifier to predict meaningful aleatoric and epistemic uncertainties.

Note that contrary to the architecture of the UQGAN described in Fig. 6.3, we do not need a cAE on the toy examples as these are already in a low-dimensional space. Thus, we can consider the encoder and decoder parts in Fig. 6.3 to be an identity mapping in this case.

8.4.2 *UQGAN Out-of-Class examples*

One of the advantages that is setting the UQGAN apart from other methods is the ability to generate its own auxiliary OoC data. In this section we will take a look at what these samples actually look like and if we can spot some patterns. As listed in Table 8.2, we have 4 different ID datasets. On each of these datasets we train a UQGAN model and use the cAE and generator to sample OoC data. For the used models please consult Section 8.2.

Starting with the results on the MNIST 0-4 dataset (Fig. 8.4 (a)), we can see that most of the generated examples are semantically similar to their respective class. However, clear distortions and other artifacts can be observed. For the class 0 the generator mostly produces samples where the zero circle is not closed, or individual strokes are disconnected. Similar artifacts can be observed for the other classes such as 2, where some samples look similar to the digits 7 and 3. Some generated OoC examples look realistic. Despite these visual similarities, distinguishing between real and generated data is still possible.

Switching to CIFAR10 0-4 (Fig. 8.4 (b)), the interpretation of the OoC examples becomes increasingly difficult. As the CIFAR10 dataset contains natural images in RGB format, the diversity is much higher than for the MNIST dataset. The displayed

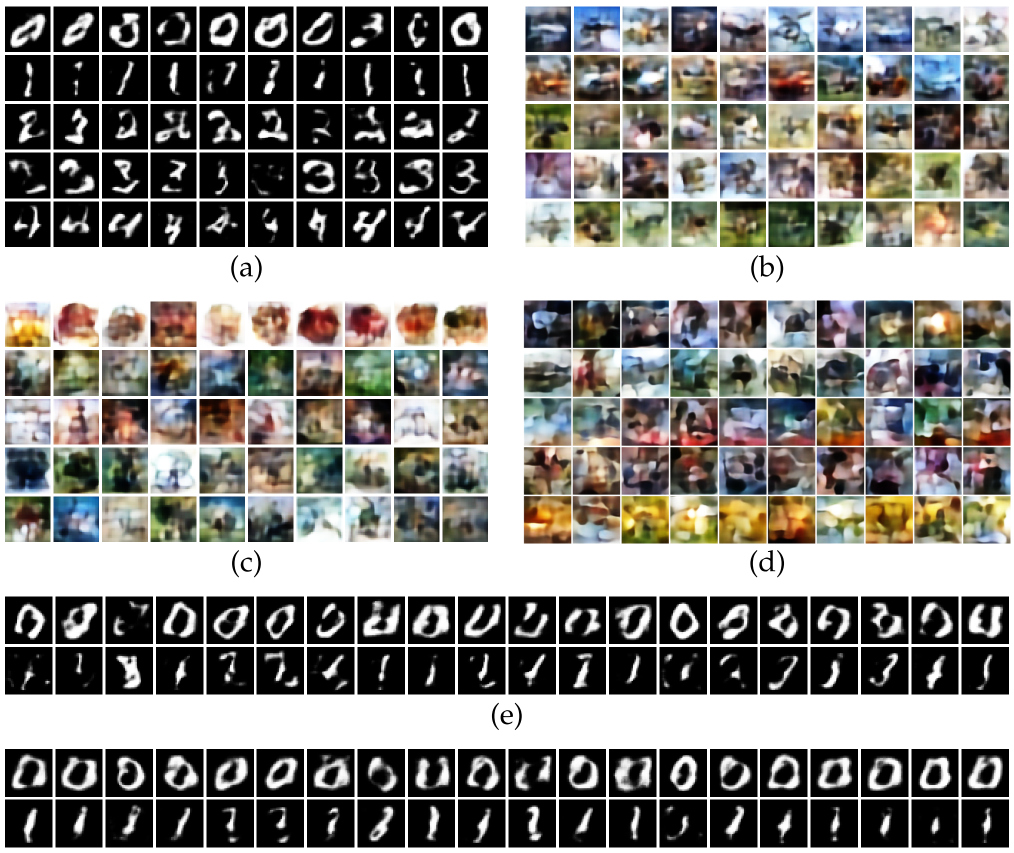

Figure 8.4: Out-of-Class examples generated by UQGAN trained on MNIST 0-4 (a), CIFAR10 0-4 (b), CIFAR100 0-49 (c) and Tiny ImageNet 0-99 (d) datasets. For the CIFAR100 0-49 and Tiny ImageNet 0-99 only the first 5 classes are displayed. Out-of-Class examples for $\lambda_R = 32$ (e) and $\lambda_R = 0$ (f) are displayed for the digits 0 and 1. Rows contain different classes and columns are multiple examples of the same class.

classes are *airplane*, *automobile*, *bird*, *cat* and *deer* (from top to bottom). Having this information at hand allows spotting superficial semantics. In the top row we can see a lot of blue and white colors which are usually seen in the background of airplane images. Although it is not easy to recognize objects on images of the size 32×32, the second row contains some parts which look similar to car engine hoods and tires.

It becomes even more difficult to interpret the OoC data for CIFAR100 0-49 and Tiny ImageNet 0-99 due to the larger number of classes. For CIFAR100 0-49 the classes *apple*, *aquarium fish*, *baby*, *bear* and *beaver* are shown (Fig. 8.4 (c)), while *Egyptian cat*, *reel*, *volleyball*, *rocking chair* and *lemon* are displayed for Tiny ImageNet 0-99 (Fig. 8.4 (d)). Images in the top row of Fig. 8.4 (c) look quite similar to an apple while the ones in the bottom row of (d) resemble the same colors as a lemon.

The influence of the low-dimensional regularizer on the generated OoC examples can be observed in Fig. 8.4, for $\lambda_R = 32$ in (e) and $\lambda_R = 0$ in (f). There, we can notice

Method	Abbreviation	Category
Proposed Approaches		
Gradient Metrics (Chapter 5)	GCE	Frequentist (External)
UQGAN (Chapter 6)	UQGAN	Generative
UQGAN with MCD (Chapter 6)	UQGAN-MCD	Generative
Other Methods		
Maximum Class Probability [52]	MCP	Frequentist (External)
Prediction Entropy (Eq. (2.13))	$\tilde{H}$	Frequentist (External)
Gradient Norm on KLD [60]	GKLD	Frequentist (External)
One-versus-All (Section 6.1)	OvA	Frequentist (Internal)
Monte Carlo-Dropout [37]	MCD	Bayesian
Bayes-by-Backprop [9]	BBB	Bayesian
Deep Ensembles [75]	DE	Ensemble
Confident Classifier GAN [79]	CCGAN	Generative
Evidence GAN [117]	EGAN	Generative
Prediction Entropy Oracle (Eq. (2.13))	$\tilde{H}$-O	Oracle
One-versus-All Oracle (Section 6.1)	Ova-O	Oracle

Table 8.3: Related methods used as benchmarks in the quantitative evaluation. All methods were discussed in Chapter 4. Abbreviations will be used in tables and figures.

that the visual diversity with the regularizer ($\lambda_R = 32$) is much higher compared to the examples without it ($\lambda_R = 0$). Particularly for the digit 1, there is a noticeable difference, where in some cases the generated examples for $\lambda_R = 32$ look similar to the digits 3 and 2. Similar observations can be made for the same digit with $\lambda_R = 0$, but much less pronounced.

8.5 QUANTITATIVE EVALUATION

8.5.1 *Benchmarking Methods*

We will compare the performance of the proposed gradient metrics and UQGAN methods to a variety of related approaches. In Chapter 4 we grouped UQ approaches into four categories: Frequentist (Internal/External), Bayesian, Ensemble and Generative. To have a diverse benchmarking suite, every aforementioned category should be included for comparison. Table 8.3 shows an overview of methods which we will compare against.

From the category of Frequentist methods we will include the MCP by Hendrycks and Gimpel [52] and the entropy over the predictive class distribution. Both approaches act as a baseline, with the entropy being often slightly better than MCP since the entire predicted distribution is considered instead of only the class of the highest confidence. Additionally, the L_1-norm of the gradient with respect to the KLD between the predicted and a uniform class distribution, as proposed by Huang, Geng, and Li [60], is considered as a comparison to the gradient metrics discussed in Chapter 5. Since the above three methods are all external approaches, we will also include the OvA approach discussed in Section 6.1, but without generated OoC data. This will give us insights about the influence of the generated samples on the model performance.

MCD by Gal and Ghahramani [37] and Bayes-by-Backprop (BBB) by Blundell et al. [9] will be chosen as representatives for Bayesian methods. Both deliver theoretically grounded uncertainties and as we are also integrating MCD into the UQGAN model, the pure MCD approach on an NN trained on the cross-entropy is an important baseline to identify the performance gain by the UQGAN architecture.

The Deep Ensemble (DE) method by Lakshminarayanan, Pritzel, and Blundell [75] is competing for the class of ensemble methods. Although the cost of training ensembles is usually higher compared to, e.g., MCD, DEs usually achieve better performance on several tasks.

Two methods utilizing GANs to generate auxiliary data are included to identify advantages in generation capabilities of our UQGAN architecture. Lee et al. [79] train a softmax based classifier and maximize the predictive entropy on the generated OoD examples. Sensoy et al. [117] on the other hand, employ the theory of evidence and combine this with a similar architecture as the UQGAN, learning the GAN inside a low-dimensional latent space. The differences of the approaches to the proposed UQGAN are discussed in the beginning of Chapter 6.

As an upper bound on the OoD detection performance, we will also include two networks which will have access to data from the datasets which are part of the OoD domain, as summarized in Table 8.2. In practical applications, access to this data is usually not given, which is why these variants are called "oracles". One of them is trained with a softmax output and cross-entropy on the ID data, while maximizing entropy on the given OoD data (similar to [79]). The second one is similarly trained as the classifier of the UQGAN but instead of receiving OoC examples from the GAN, the given OoD data is presented to the classifier. Both will give a reference for what is achievable if we have perfect knowledge about the OoD data. Additionally, the OvA oracle is giving insights on how well our generated OoC data shields the ID from the real OoD data and whether generated data might be even better for generalization than real OoD examples.

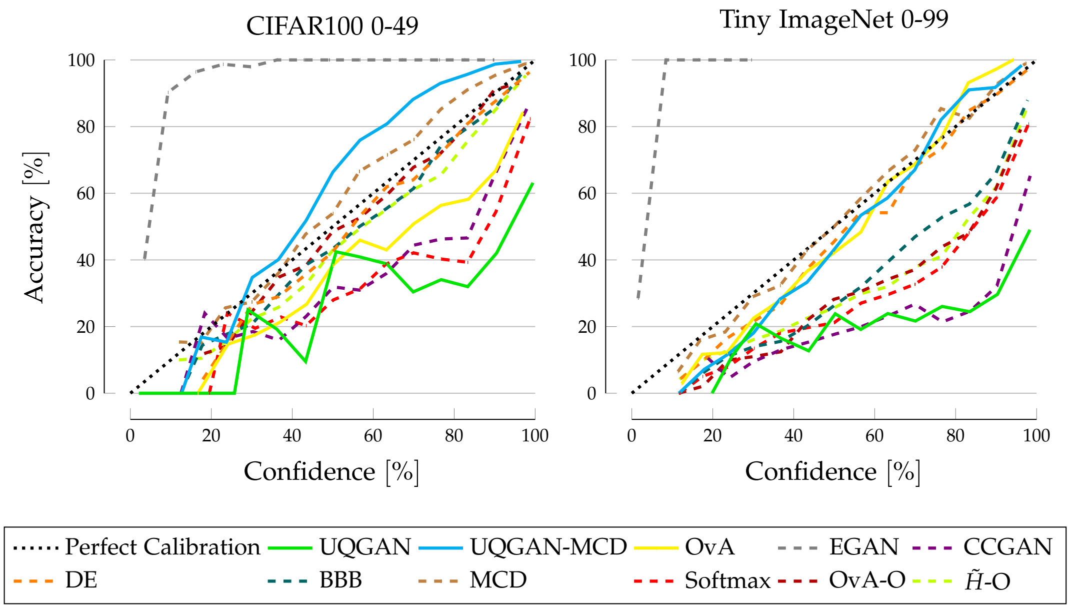

Figure 8.5: Calibration plots for all methods from Table 8.3. Note that MCP, $\tilde{H}$, GCE and GKLD are external methods which all operate on a softmax based classifier trained with the cross-entropy on the same dataset. Therefore, this network is only shown once in the figure (Softmax). Methods from this work are marked with a solid line.

MCP, predictive entropy, GKLD, MCD, DEs, CCGAN, entropy oracle and the gradient metrics from Chapter 5 all use a classifier with softmax output which is trained on the cross-entropy loss.

In the following sections we will use the abbreviations of the related methods for brevity, as specified in Table 8.3.

8.5.2 *Classification Accuracy and Calibration*

This section will analyze the effects of the approaches from Table 8.3 on the calibration and classification accuracy.

Starting with the calibration, Fig. 8.5 shows calibration curves on the CIFAR100 0-49 and Tiny ImageNet 0-99 datasets, while Table 8.4 shows the ECE on the two mentioned datasets. What is most noticeable in the graphic is the under-confidence of the EGAN method, which has a classification accuracy of 100 % from a confidence of 40 % upwards on the CIFAR100 0-49 dataset. For the Tiny ImageNet 0-99 dataset EGAN has a lot of empty bins and the calibration curve is only populated in the $[0\,\%, 30\,\%]$ confidence interval. This is also captured by the ECE, where the EGAN has the highest and third highest Expected Calibration Error on the CIFAR100 0-49 and Tiny ImageNet 0-99 datasets, respectively. On the other end of the calibration curves we have the UQGAN, CCGAN and softmax baseline as the methods with the

Method	CIFAR100 0-49			Tiny ImageNet 0-99		
	Accuracy ↑	AUROC S/F ↑	ECE ↓	Accuracy ↑	AUROC S/F ↑	ECE ↓
GCE	56.12 (0.60)	79.17 (0.28)	19.10 (3.23)	36.06 (0.30)	76.18 (0.88)	26.01 (7.25)
UQGAN	56.60 (0.73)	73.61 (0.33)	34.32 (0.42)	34.28 (0.37)	**71.90 (0.47)**	**48.94 (0.66)**
UQGAN-MCD	**64.53 (0.47)**	80.38 (0.40)	10.92 (0.24)	**45.60 (0.43)**	79.18 (0.42)	5.92 (0.38)
MCP	56.12 (0.60)	80.68 (0.29)	19.10 (3.23)	36.06 (0.30)	78.56 (0.68)	26.01 (7.25)
$\tilde{H}$	56.12 (0.60)	81.16 (0.26)	19.10 (3.23)	36.06 (0.30)	79.39 (0.74)	26.01 (7.25)
GKLD	56.12 (0.60)	**72.69 (1.16)**	19.10 (3.23)	36.06 (0.30)	77.96 (0.72)	26.01 (7.25)
OvA	**50.90 (0.83)**	76.21 (0.91)	23.89 (2.16)	35.18 (0.26)	76.23 (0.43)	11.59 (4.41)
MCD	59.88 (0.81)	82.57 (0.60)	21.94 (0.58)	43.48 (0.53)	80.63 (0.30)	**2.79 (0.35)**
BBB	56.02 (0.50)	81.90 (0.55)	14.71 (0.31)	32.31 (0.43)	78.44 (0.91)	19.39 (0.62)
DE	62.36 (0.43)	**82.77 (0.47)**	**2.72 (0.37)**	42.48 (0.22)	**81.29 (0.38)**	16.50 (6.94)
CCGAN	54.16 (0.13)	81.07 (0.44)	27.19 (5.45)	36.07 (0.53)	78.66 (0.67)	35.48 (2.13)
EGAN	51.16 (0.26)	77.83 (0.80)	**41.26 (2.05)**	**30.54 (0.84)**	73.40 (0.91)	28.79 (0.79)
$\tilde{H}$-O	53.66 (0.39)	81.39 (0.69)	19.82 (0.34)	37.18 (0.50)	79.50 (0.63)	17.05 (2.82)
OvA-O	51.82 (1.81)	76.43 (0.83)	15.26 (4.27)	34.92 (0.56)	75.10 (0.92)	20.61 (3.50)

Table 8.4: Accuracy, AUROC and ECE for the CIFAR100 0-49 and Tiny ImageNet 0-99 datasets. AUROC is computed on the task of FP detection (success/failure). For each evaluation metric the best result is highlighted in bold green and the worst in bold red. Please have a look at Table A.1 for the results on the MNIST 0-4 and CIFAR10 0-4 datasets.

most over-confidence. We can see that the OvA classifier is considerably closer to the perfect calibration line than the UQGAN, which suggests that the generated OoC data has a negative effect on the calibration of the model. Additionally, the OvA oracle has also a lower ECE than the standard UQGAN, putting more weight on the hypothesis that the generated data increases the calibration error. When analyzing the results for the UQGAN-MCD, we can see a considerable improvement over the standard UQGAN. Not only is the ECE consistent within the top-2 of all methods, but on the CIFAR100 0-49 dataset the error shifted from an over- to an under-confidence. Generally, DE and MCD are performing well over both datasets, with DE being slightly under-confident and MCD over-confident. The ECE results on the MNIST 0-4 and CIFAR10 0-4 datasets can be found in Appendix A (Table A.1). There, the conclusions are similar to the ones described above.

Switching our focus to the classification accuracy in Table 8.4, we can see that the UQGAN-MCD is clearly outperforming other methods. On CIFAR100 0-49 a classification accuracy of 64.53 % is only a relative improvement of 3.5 % over the second-best competitor (DE) but comparing with closely related methods like CCGAN and EGAN, we achieve a relative improvement of 26.1 %. The gap becomes larger when evaluating on Tiny ImageNet 0-99, where UQGAN-MCD is relatively improving by 4.9 % and 49.3 % over MCD (second-best) and EGAN (worst). This also shows that the class shielding with OoC data can also improve the classification accuracy, which is only possible due to the OvA framework. When comparing the standard UQGAN architecture, we only achieve mid-tier results compared to the

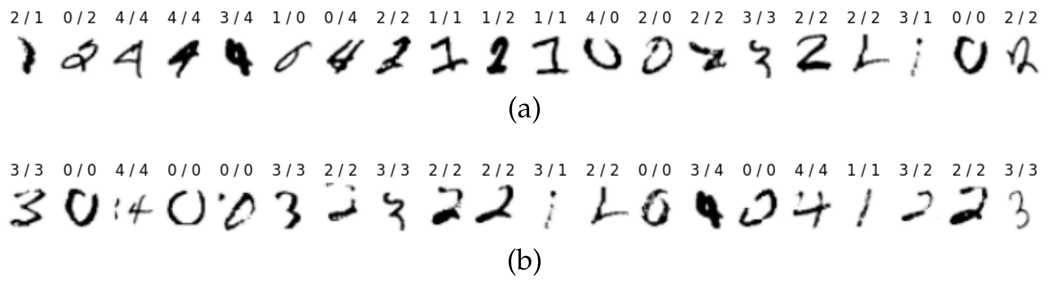

Figure 8.6: Top 20 samples from the MNIST 0-4 test set with the highest aleatoric (a) and epistemic (b) uncertainty predicted by a UQGAN. Numbers over each digit correspond to *prediction/annotation*.

related works, being surpassed by DE and MCD on CIFAR100 0-49 and only improving over BBB and EGAN on Tiny ImageNet 0-99. Again, similar results can be observed on the MNIST 0-4 and CIFAR10 0-4 datasets, which can be found in Appendix A (Table A.1). For quantitative evaluations, MNIST is difficult to interpret as all methods achieve classification accuracies of greater than 99 %, similarly for the ECE. On CIFAR10 0-4 we again achieve relative improvements of 5.2 % and 9.5 % over MCD (second-best) and MCP (worst) when considering accuracy. For the ECE, UQGAN is again producing mid-tier results while UQGAN-MCD is best on CIFAR10 0-4. What is interesting to observe is that UQGAN as well as UQGAN-MCD have a higher classification accuracy than OvA-O. This means that GAN generated OoC data can help improve generalization (in terms of classification accuracy), while real OoD data might be too far away from the ID. Gradient metrics will not be covered in this section as the approach is external and does not influence the training of the classification model.

The results for the UQGAN variants in this section in comparison to the related work can be summarized as the following:

- UQGAN achieves mid-tier results in classification accuracy and low-tier results on the calibration error
- UQGAN-MCD improves considerably and achieves top-2 results on ECE while being best with regard to classification accuracy
- Generated OoC examples have a negative influence on the ECE of both models
- In comparison to closely related methods (CCGAN, EGAN) we considerably better results with large margins to the second-best competitor

8.5.3 *False Positive Detection*

In this section we will quantify the ability of UQ methods to detect misclassifications, which are also called False Positives (FPs), when considering the predicted class as the positive class. Figure 8.6 shows the top-20 MNIST 0-4 test set examples with respect to the highest aleatoric uncertainty (a) and highest epistemic uncertainty (b) predicted by a UQGAN. As we can see, among the examples with high aleatoric uncertainty we have 9 misclassified digits. Considering that in this case the classification accuracy of the model was 99.68 % and that the MNIST 0-4 test set has 5000 samples, the 20 displayed digits already contain 56.25 % percent of all misclassified test examples. Comparing the epistemic uncertainty against that, we only find 3 misclassifications. This is a meaningful observation, as FPs usually arise in overlapping class regions with high data uncertainty, which is part of the aleatoric uncertainty. Following this reasoning, we will consider the aleatoric uncertainty for approaches that are able to distinguish between the different types. For methods which are not able to distinguish, we will use the uncertainty/confidence measure defined by the method.

In Table 8.4 in the AUROC S/F columns you can see the results on the task of FP detection for the CIFAR100 0-49 and Tiny ImageNet 0-99 datasets. Note that the same table for MNIST 0-4 and CIFAR10 0-4 can be found in Appendix A. Bayesian (BBB, MCD) and ensemble methods perform best on this task, with DE mostly being the best performing one. This underlines the importance of Bayesian/sampling approaches for ID reliability. The UQGAN is only achieving low to mid-tier results, while the UQGAN-MCD is considerably better but still not among the best, except for CIFAR10 0-4 where both are obtaining comparatively high AUROC S/F results. Comparing the two gradient metric approaches GCE and GKLD, GCE is usually better (except for the Tiny ImageNet 0-99 dataset). It is noteworthy that GKLD has especially low AUROC S/F values on datasets with a high classification accuracy such as MNIST 0-4 and CIFAR10 0-4 (see Table A.1). This could be explained by the fact that with a high accuracy the classification model becomes more confident, also increasing the confidence on unknown inputs. As the GKLD is using the KLD between the prediction and a uniform class distribution, the loss increases exponentially the more confident the prediction is, which is amplified by considering the whole predictive distribution. This makes it especially difficult to distinguish ID from Out-of-Distribution inputs. Both approaches are inferior to many of the related methods, which is expected on this task as they are aimed at quantifying the model uncertainty. The close to the UQGAN related methods CCGAN and EGAN are on most of the four datasets ranking between UQGAN and UQGAN-MCD. An important aspect is that the difficulty of the FP detection task increases with classification accuracy. Since the UQGAN-MCD has a much higher accuracy for most datasets, the comparisons for the FP detection task should be treated with caution.

Method	CIFAR100 0-49			Tiny ImageNet 0-99		
	AUROC ↑	AUPR-In ↑	FPR95 ↓	AUROC ↑	AUPR-In ↑	FPR95 ↓
GCE	65.11 (0.68)	18.83 (0.53)	88.99 (1.11)	65.81 (2.68)	22.10 (1.69)	83.85 (3.30)
UQGAN	80.11 (1.40)	28.81 (1.32)	**55.23 (3.20)**	79.25 (1.61)	26.35 (2.25)	47.22 (1.85)
UQGAN-MCD	**80.75 (1.19)**	**31.75 (1.53)**	58.10 (2.11)	**94.96 (0.13)**	**59.76 (0.64)**	**13.72 (0.30)**
MCP	67.68 (1.56)	23.03 (1.71)	88.42 (1.42)	61.53 (1.04)	21.07 (0.98)	92.51 (0.87)
$\tilde{H}$	69.43 (1.71)	23.75 (1.85)	87.08 (1.74)	62.44 (1.31)	21.90 (0.86)	93.79 (1.28)
GKLD	69.82 (1.29)	15.74 (1.33)	82.82 (1.85)	66.01 (2.38)	21.23 (3.03)	91.56 (2.16)
OvA	**62.99 (1.59)**	**15.69 (2.69)**	**92.44 (0.76)**	**55.19 (2.29)**	**17.97 (2.41)**	**97.32 (0.62)**
MCD	67.75 (1.15)	22.31 (1.14)	89.38 (1.30)	63.35 (4.23)	27.11 (2.29)	95.86 (1.28)
BBB	69.74 (0.76)	24.60 (1.25)	87.01 (0.75)	68.05 (2.29)	21.24 (2.14)	81.70 (3.53)
DE	74.29 (0.50)	29.38 (0.67)	83.37 (1.41)	67.76 (0.27)	30.85 (0.42)	93.79 (0.23)
CCGAN	68.66 (0.48)	22.51 (0.20)	86.17 (0.55)	59.99 (1.84)	20.58 (1.61)	94.49 (0.51)
EGAN	77.43 (2.58)	26.12 (2.41)	61.48 (4.33)	84.65 (5.42)	36.86 (8.25)	39.67 (12.50)
$\tilde{H} - O$	86.16 (0.41)	38.63 (0.58)	44.67 (1.46)	85.43 (1.90)	41.95 (2.31)	48.74 (6.63)
OvA-O	92.91 (0.28)	47.68 (1.37)	22.35 (0.71)	95.75 (0.09)	59.69 (0.50)	10.30 (0.46)

Table 8.5: Results on the task of OoD detection measured with the AUROC, AUPR-In and FPR95 evaluation metrics. For CIFAR100 0-49 the OoD set is constructed using {CIFAR100 50-99, LSUN, SVHN, Fashion-MNIST, MNIST} and for Tiny ImageNet 0-99 we take {Tiny ImageNet 100-199, SVHN, Fashion-MNIST, MNIST}, as described in Table 8.2. For each evaluatino metric and dataset the best result is highlighted in bold green and the worst in bold red. Please have a look at Table A.2 for the results on the MNIST 0-4 and CIFAR10 0-4 datasets and Tables A.3 to A.6 for a breakdown of the results between individual datasets.

The main takeaways from this section are

- Bayesian and ensemble methods are best-in-class for the task of FP detection
- UQGAN is ranking in a low-tier compared to other approaches
- UQGAN-MCD achieves considerable higher FP detection rates but is still surpassed by Bayesian methods
- GCE is outperforming GKLD on three of the four datasets

8.5.4 *Out-of-Distribution Detection*

Out-of-Distribution detection is the task of identifying inputs which do not stem from the same distribution as the training data. Due to a large variability in the real world and the high number of parameters in DNNs (causing model uncertainty), OoD examples are a common phenomenon and a large threat to NNs in the wild. In this section we will analyze the ability of different UQ methods to detect such inputs based on their defined uncertainty/confidence score.

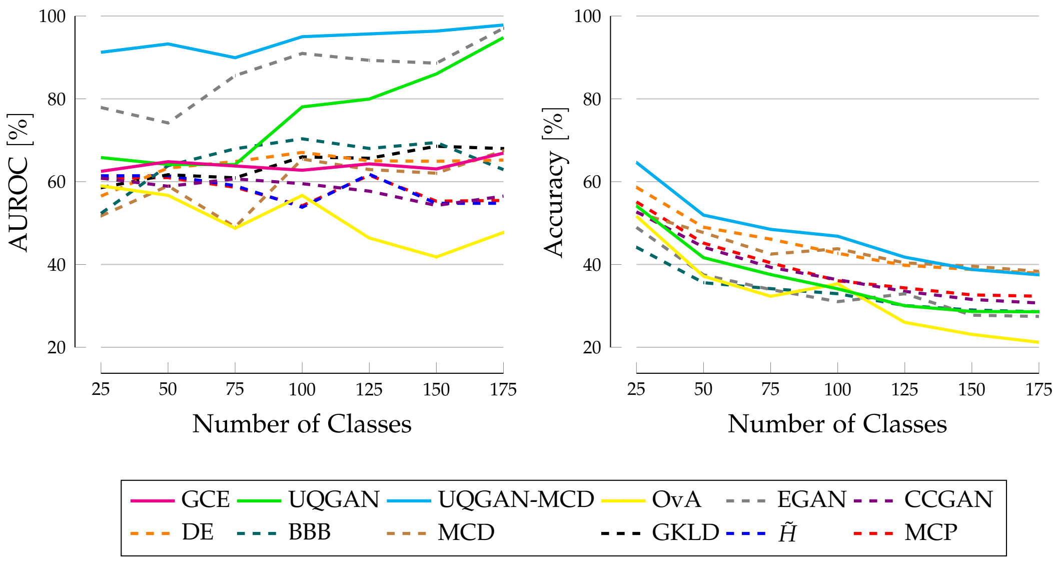

Figure 8.7: AUROC for the task of OoD detection and accuracy as a function of the number of classes on the Tiny ImageNet dataset. Number of classes corresponds to the ID set, while the complement of classes is part of the OoD set alongside the datasets mentioned in Table 8.2. Note that in the accuracy plot only the MCP baseline is displayed as the GCE, GKLD and $\tilde{H}$ are external methods and use the same weights. Methods from this work are marked with a solid line.

Table 8.5 summarizes the results on the CIFAR100 0-49 and Tiny ImageNet 0-99 datasets. The results on MNIST 0-4 and CIFAR10 0-4 can be found in Appendix A (Table A.2). Most noticeable is that the UQGAN-MCD approach is outperforming other methods on all four datasets, especially when considering the FPR95 metric. On CIFAR100 0-49 UQGAN-MCD achieves a relative improvement of 3.9 %, 8.1 % and 5.5 % in AUROC, AUPR-In and FPR95, respectively, to the second-best competitor. This increases on the Tiny ImageNet 0-99 dataset to 10.3 %, 62.1 % and 65.4 % relative improvement in the three aforementioned performance metrics. Similar large margins can be observed on the MNIST 0-4 and CIFAR10 0-4 datasets. Interestingly, one can observe two groups when considering the FPR95 evaluation metric. The first group, containing UQGAN, UQGAN-MCD and EGAN, achieves results with very large margins to the second group, containing all other methods. This suggests that utilizing GAN generated auxiliary data in a low-dimensional space is crucial for the FPR95 performance measure, which is highly important for practical applications. The importance of GAN generated data is also enforced by the inferior results of the OvA baseline, which does not receive any additional data. GAN generated data alone does not improve the results considerably, as can be observed with the CCGAN method. Another factor which distinguishes the first group from the second are the OoD detection results for different number of classes. Figure 8.7 visualizes the influence of the number of classes on the Tiny ImageNet dataset. For this, an increasingly large subset of the dataset is used as ID, while the complement serves as the usual OoD data. All models and hyperparameters were kept fixed for this experiment. The most apparent result is that the accuracy decreases with the number of classes. This is a reasonable result as we kept the classification model fixed to a ResNet 18. More interesting is the fact that both UQGAN models and the EGAN approach have a higher AUROC for a higher number of classes. As all other methods have a constant or degenerating performance, this can again be attributed to the low-dimensional GAN training and the usage of an AE for dimensionality reduction. One hypothesis for this could be that the AE is learning more diverse feature embeddings when the number of classes, and thus also the number of training examples, increases. More gains could be realized with advanced training of the AE, e.g., in terms of pre-training on large scale datasets. Comparing the results of the UQGAN to the other generative methods we can see that both, CCGAN and EGAN, are outperformed by UQGAN as well as UQGAN-MCD on almost all datasets. Only on Tiny ImageNet 0-99 the UQGAN is surpassed by EGAN with UQGAN-MCD still being best with a large margin. Tables A.3 to A.6 are breaking down the results from Tables 8.5 and A.2 between the respective ID sets and all individual OoD datasets. Evaluating the results for the UQGAN approaches it becomes evident that UQGAN-MCD is particularly good on hard OoD examples (the second half of the ID class-wise split) over all four datasets. While not always best with respect to all dataset configurations, UQGAN-MCD is surpassing all other approaches on the higher dimensional Tiny ImageNet 0-99 dataset. On the same dataset, the gap between UQGAN and UQGAN-MCD is especially large when considering FMNIST and MNIST as OoD datasets. Although we would consider these two datasets to be easily differentiable from Tiny ImageNet images, this seems to be not the case as the other competitors are also struggling on this task. Only the proposed UQGAN-MCD is achieving overall good results, outperforming the entropy oracle on all Tiny ImageNet comparisons. This result indicates that both model uncertainty based on a sampling approach and generative auxiliary data play an important role in OoD detection. The UQGAN variant is most of the time ranking mid- to top-tier in the comparisons on the individual datasets. When analyzing the results of the oracles in relation to the UQGAN two things become apparent. First, there often remains a lot of room for improvement, especially with respect to the FPR95 evaluation metric. Second, the entropy oracle is far weaker than the OvA oracle on the higher dimensional CIFAR100 0-49 and Tiny ImageNet 0-99 datasets, even being surpassed by the UQGAN-MCD on Tiny ImageNet 0-99. An implication of this is that the entropy maximization, as done in [53, 79], is not sufficient for generalization to new unknown data, an observation which was also made by Vernekar et al. [136]. The authors of [136] are suggesting to use a reject classifier instead (a classifier including a $K+1$-st class which can be used to reject inputs). For the OvA approach, as used in the other oracle as well as both UQGAN models, each class learns its own reject classifier. The final reject decision is then build by aggregating the individual reject classifiers with Eq. (6.21). The fact that the OvA oracle does perform considerably better compared to the entropy maximization oracle strongly suggests that the assumption of Vernekar et al. [136] could be true.

Switching our attention to the gradient metrics, we can see that GKLD is slightly better than GCE in terms of AUROC, but the results of AUPR-In and FPR95 are not as clear. On CIFAR100 0-49 GKLD has a 6.9 % relative improvement in FPR95 over GCE, while on Tiny ImageNet 0-99 GCE is relatively improving by 8.4 %. For MNIST 0-4 and CIFAR10 0-4 we have a similar situation as in Section 8.5.3, where GKLD is achieving very poor results. Again, this can be attributed to the generally higher confidence levels on less complex tasks. Compared to the other methods, gradient metrics can only compete on the higher dimensional problems CIFAR100 0-49 and Tiny ImageNet 0-99. With regard to MNIST 0-4 and CIFAR10 0-4 both, GCE and GKLD, are marginally worse than the baselines MCP and $\tilde{H}$. Considering the comparisons between individual datasets in Tables A.3 to A.6, we can observe that GKLD is almost always worse than GCE with respect to the difficult OoD examples (the second half of the ID class-wise split). For the more distinct OoD inputs, GKLD is generally obtaining better results than GCE (except for MNIST 0-4). This result fits into the previous observations that GKLD suffers from high confidence predictions, which are more likely to occur on data which is very close to the ID and less likely for more distinct OoD examples. Overall, GCE ranks mid- to low-tier on most dataset specific benchmarks.

Summarizing the insights of this section we can say that

- UQGAN as well as UQGAN-MCD are achieving state-of-the-art results in comparison to previous works on GAN based OoD detection as well as in comparison to methods from other categories
- On the FPR95 evaluation metric, which is especially important for practical applications, UQGAN-MCD is outperforming others methods with large margins
- Entropy maximization is not sufficient for generalization to unknown OoD data
- There is no clear evidence as to which of the gradient approaches, GCE or GKLD, is better suited for OoD detection
- Gradient metrics only have an advantage on higher dimensional problems while being surpassed by the baselines on the very small MNIST 0-4 and CIFAR10 0-4 datasets

8.5.5 *Adversarial Example Detection*

Adversarial Examples can be a large threat for DNNs when applied in real world scenarios. Neural Networks might be victims of malicious attacks in, e.g., autonomous driving, surveillance or malware and financial fraud detection. Figure 8.8 shows examples for the FGSM and PGD Least-Likely attacks on an image from the Tiny ImageNet 0-99 dataset for various levels of perturbation strength. For small values of ϵ the perturbation is nearly imperceptible, while adversarial examples

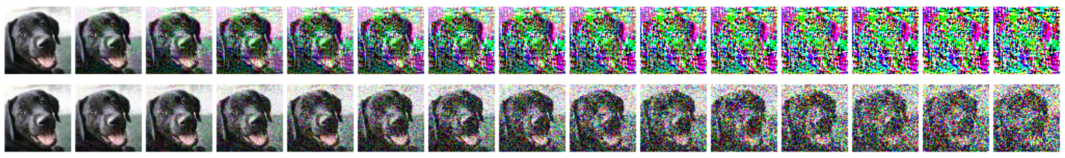

Figure 8.8: FGSM [43] (top) and PGD [88] Least-Likely (bottom) attacks for varying levels of perturbation strength ϵ from 0 (left) to 1 (right). All attacks were made on a UQGAN which was trained on the Tiny ImageNet 0-99 dataset. The image shown is a test sample from the aforementioned dataset.

with larger perturbation strengths can clearly be distinguished from real images. A noticeable difference is that FGSM produces much stronger perturbations than PGD, showing the advantage of iterative approaches. While humans can easily identify adversarial examples with extreme perturbations, the same does not generally hold for DNNs. This is especially important in situations where the actual content analyzed by the NN can be altered freely as is the case for images[1]. For someone with a malicious intent, a targeted attack would be the methodology of choice. An attack, for example in the context of automated driving, could be carried out by placing a sticker on a stop sign, which fools the network to detect a speed limit sign [34].

In this section we will analyze to which extent UQ methods are capable of detecting such attacks. If possible, adversarial examples could be filtered out together with FPs and OoD examples.

Choice of Uncertainty Type

One of the first questions that comes up when trying to perform adversarial example detection based on uncertainties is the type of uncertainty which should be used. Smith and Gal [123] argue that adversarial examples lie off the natural image manifold and therefore can be detected using epistemic uncertainty estimates. Other works (e.g., [24, 44]) show results in favor of the aleatoric uncertainty or a mixture of epistemic and aleatoric. To gain some insights into the choice of uncertainty types, we evaluate the UQGAN and UQGAN-MCD using the three aforementioned attacks and different uncertainties. Figure 8.9 shows the results for different amounts of perturbation strength on the Tiny ImageNet 0-99 dataset, measured with the AUROC (for adversarial example detection) and success rate (fraction of samples which have a different predicted label after the attack). First, it is noticeable that the PGD approach achieves higher success rates over a wider range of ϵ values than FGSM. Regarding the detection, aleatoric uncertainty is obtaining higher AUROC results for the untargeted attacks (FGSM and PGD). This is especially apparent for the PGD approach, where aleatoric uncertainty and epistemic uncertainty have

1 In the case of, e.g., malware applications, which are altering their code in order to represent an adversarial example, the content of the source code needs to be preserved in order to maintain functionality. In this case large values of perturbation strength are likely not possible.

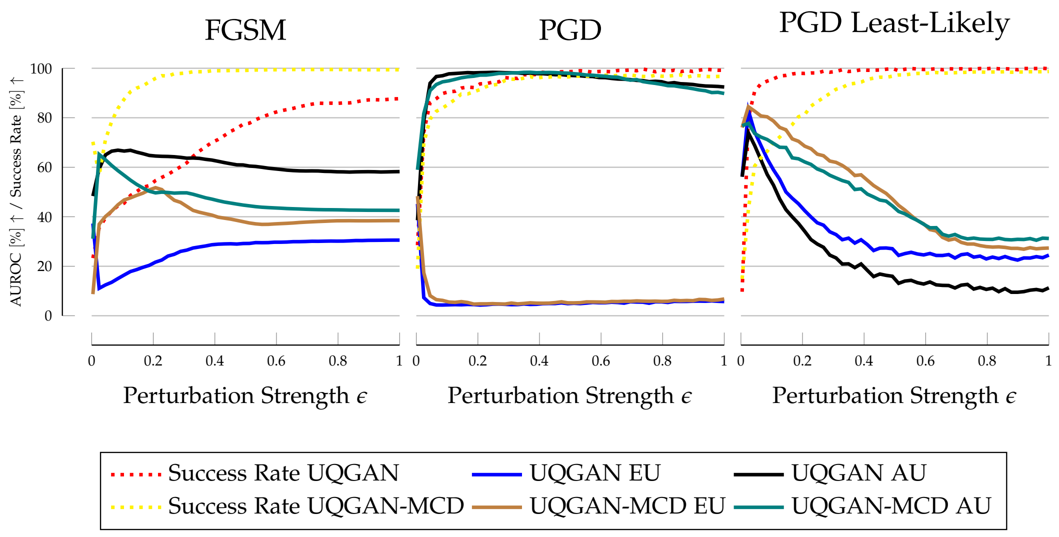

Figure 8.9: Success and detection rates on the UQGAN and UQGAN-MCD models trained on the Tiny ImageNet 0-99 dataset. Adversarial examples are generated by the FGSM [43] (left), PGD [88] (middle) and PGD Least-Likely (right) with a white-box attack. For each model the aleatoric as well as the epistemic uncertainties are considered for adversarial example detection. For the results on the MNIST 0-4, CIFAR10 0-4 and CIFAR100 0-49 datasets please see Fig. A.1.

an absolute gap of approximately 80 %. For PGD Least-Likely, which targets the least likely class, the epistemic uncertainty is achieving better results compared to the aleatoric uncertainty. An explanation for this could be the fact that in the case of targeted attacks the decision boundary of the target class might be far away compared to other classes, which forces the attacked image to leave the ID domain. In contrast to this, untargeted attacks can generate perturbations which move the adversarial example to the closest decision boundary, as these require less perturbation. The resulting adversarial examples are located in ID regions with high data uncertainty, explaining the large advantage of aleatoric uncertainty in this setting. It suggests that in practice a combination of aleatoric and epistemic uncertainty is required to detect all kinds of attacks, which is in line with the findings of Grosse et al. [44]. This result also supports the two-stage approach of the UQGAN method, where in practice, aleatoric and epistemic uncertainty need to be considered jointly. Results on the MNIST 0-4, CIFAR10 0-4 and CIFAR100 0-49 datasets can be found in Appendix A (Fig. A.1). On these datasets, the results are coherent with the ones on the Tiny ImageNet 0-99 dataset, with aleatoric uncertainty being better for detecting untargeted attacks and epistemic uncertainty for targeted attacks. Therefore, in the following we will evaluate the aleatoric uncertainty on untargeted attacks and the epistemic uncertainty on targeted adversarial examples. Methods which do not distinguish into different sources of uncertainty, will use their defined confidence/uncertainty measure for both attack types.

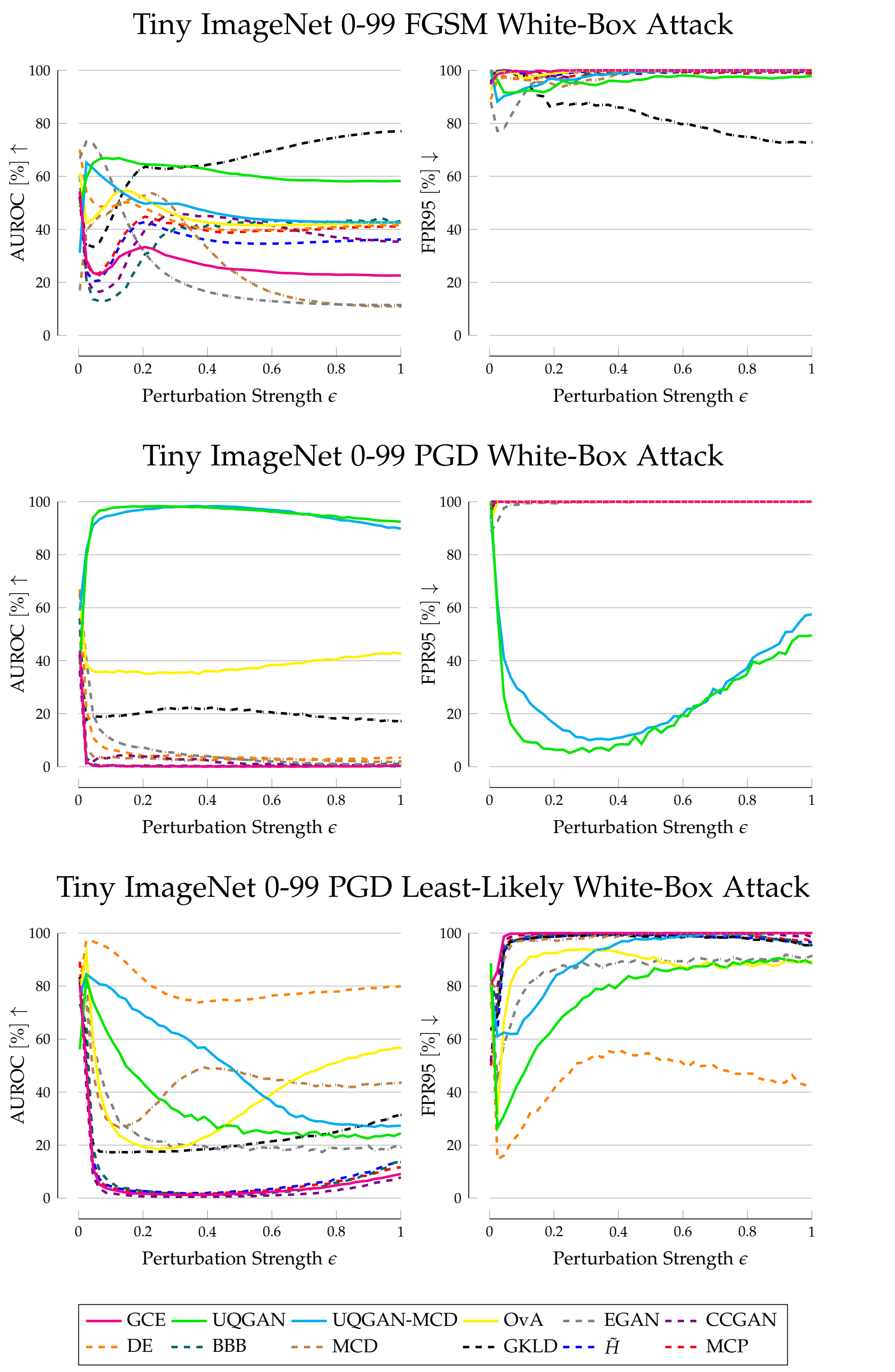

Figure 8.10: Detection rates of adversarial examples generated by the FGSM [43] (top), PGD [88] (middle) and PGD Least-Likely (bottom) with a white-box attack on the Tiny ImageNet 0-99 dataset. Evaluation metrics are AUROC (left) and FPR95 (right) and are computed on the aleatoric uncertainty for the non-targeted attacks (FGSM and PGD) and on the epistemic uncertainty for the targeted PGD attack. Methods from this work are marked with a solid line. Please see Figs. A.2 to A.4 for the graphics on the MNIST 0-4, CIFAR10 0-4 and CIFAR100 0-49 datasets.

Detecting White-Box Attacks

Following our previous findings, we will now evaluate all UQ methods from Table 8.3 on the task of adversarial example detection. Figure 8.10 summarizes the results on the higher dimensional Tiny ImageNet 0-99 dataset.

Starting with the FGSM attack, we can see that the GKLD and UQGAN approaches are the only ones which have an AUROC which is consistently above 50 % for all perturbation strengths. While GKLD has an advantage for larger values of ϵ, UQGAN, UQGAN-MCD, EGAN and DE are stronger for small perturbations. Interestingly, GCE cannot compete with GKLD and most of the other methods on this task, which suggests that in order to detect adversarial examples using gradient metrics, information from all class predictions need to be taken into account. This is a reasonable result as adversarial examples are shifting the prediction probabilities from one class to another. In terms of AUROC, UQGAN-MCD and the OvA baseline are among the best when ignoring GKLD and UQGAN. EGAN has the highest AUROC and lowest FPR95 for small perturbations but is impacted strongly by larger ones.

Switching our attention to the results for the PGD attack, the most striking result is the superior performance of the UQGAN and UQGAN-MCD methods. While the AUROC for most approaches quickly vanishes to about 0 % - 5 %, the two UQGAN variants achieve an AUROC of above 80 % for small perturbations. Second, the OvA baseline, although below 50 % AUROC, achieves considerably better results than other methods and achieves AUROC values of around 40 % over all ϵ values. Ranking behind OvA is the GKLD approach at approximately 20 % AUROC for different perturbation strengths. For the FPR95 evaluation metric results are less diversified, as all methods except the UQGAN variants have an FPR95 of close to 100 %. UQGAN as well as UQGAN-MCD again achieve superior performance on this attack type, with increasing FPR95 for larger perturbation strengths, however.

On the targeted PGD Least-Likely, DE is standing out from the rest with a high AUROC and comparatively low FPR95. For low perturbation strengths of less than 0.3, the two UQGAN variants are second with the UQGAN-MCD being slightly better than UQGAN. With larger ϵ values, the AUROC of both UQGANs drops continuously and the OvA baseline and MCD are taking the second and third place starting from a perturbation strength of 0.5. The results for the FPR95 performance metric are a bit different. DE is still achieving the lowest FPR95 but the standard UQGAN is now consistently on the second place over all values of ϵ. For perturbation strengths larger than 0.5, UQGAN, EGAN and OvA baseline are obtaining the same FPR95 at around 90 %.

The results on MNIST 0-4, CIFAR10 0-4 and CIFAR100 0-49 can be found in Appendix A (Figs. A.2 to A.4). In general UQGAN is stronger than UQGAN-MCD on the PGD attack and in terms of FPR95 on the PGD Least-Likely, while the

UQGAN-MCD is better in terms of overall AUROC on PGD Least-Likely adversarial examples. For the FGSM attack, results are not always as clear between the two variants, and we cannot settle on a single best approach there. It remains however consistent over the other datasets that both UQGANs are clearly outperforming all other methods on the untargeted PGD attack. Interestingly the OvA baseline from Section 6.1 is often among the best when considering untargeted attacks. Although the results are not great, it hints on naturally higher results on the task of adversarial example detection. On the MNIST 0-4 dataset we are again observing similar results for GKLD as in the other evaluated tasks. In this case, AUROC and FPR95 values are over nearly the whole range of perturbation strength very poor. EGAN is also underperforming on MNIST 0-4, with a noticeable drop for an ϵ above 0.4. Madry et al. [88] claimed the hypothesis that PGD adversarial examples might be "universal" among first-order attacks, meaning a resistance to the PGD examples would imply a resistance to all other attacks which are utilizing first-order derivatives. As UQGAN and UQGAN-MCD have astonishing results on this type of adversarial example but are not on all datasets among the best w.r.t. the FGSM attack, this claim does not fully hold, at least when considering the task of detecting adversarial examples.

Summarizing the task of adversarial example detection, we can conclude

- There is no uncertainty type which is universally applicable for the detection of all adversarial example attacks
- GKLD is good for detecting FGSM attacks with large perturbation, GCE can not compete on any attack
- UQGAN and UQGAN-MCD achieve superior results on the untargeted PGD attack
- The OvA approach from Section 6.1 has a naturally higher detection rate of untargeted adversarial examples compared to other methods
- DEs are obtaining good results on the targeted PGD Least-Likely, followed by UQGAN, EGAN and OvA baseline in terms of FPR95

9 CONCLUSION

9.1 SUMMARY

This thesis proposed two approaches for the quantification of DNN uncertainties. Both constitute a significant contribution to the UQ community and partially achieve outstanding results.

Gradient Metrics offer a relatively inexpensive way (in terms of computational cost) to quantify model uncertainty. They do not require architectural changes and can be applied to already trained models. The quantitative results were compared against a number of other approaches and especially against a related publication (GKLD by Huang, Geng, and Li [60]) which uses a different objective to compute the gradients. On the task of FP detection the GKLD method was surpassed by GCE (proposed in Chapter 5) on three of the four datasets which were used for evaluation. Especially on the benchmarks with less complexity (MNIST 0-4, CIFAR10 0-4), GKLD was surpassed by GCE with large margins, which can be explained by the choice of loss function and the increased overall confidence of the classification models on these datasets. Compared to the other approaches, GCE ranks in a mid-tier, surpassing the baselines and some related methods on MNIST 0-4 and CIFAR10 0-4, but being inferior to most approaches on the higher dimensional CIFAR100 0-49 and Tiny ImageNet 0-99 datasets. For the task of OoD detection, results compared to other methods are reversed w.r.t. FP detection. Here, gradient metrics achieve better results on higher dimensional problems like CIFAR100 0-49 and Tiny ImageNet 0-99, while not being able to compete against many techniques on the MNIST 0-4 and CIFAR10 0-4 datasets. Between the two gradient metric methods GCE and GKLD, there is no clear evidence onto which of them is better suited for the task of OoD detection. However, it is to note that GCE has a significant advantage over GKLD on the highest dimensional Tiny ImageNet 0-99 benchmark in terms of FPR95, which is a highly important evaluation metric for practical applications. On the last downstream task, the adversarial example detection, GCE is surpassed by nearly every other method. In the experiments we argued that this is most likely due to the fact that the objective of GCE is only based on the MAP target, which is exactly the place where adversarial attacks are aiming at. In contrast to this finding, GKLD works surprisingly well for detecting untargeted attacks, especially from the FGSM method, outperforming all other approaches for large perturbation strengths. As a conclusion, gradient metrics are a suitable technique for resource constrained applications which aim to detect OoD inputs. Here, gradient metrics are outperforming other baselines and can provide a gain for practical applications in terms of FPR95.

The second proposed method, *Uncertainty Quantification GAN* (UQGAN), is able to separate aleatoric and epistemic uncertainty in a principled way. Based on a novel way to generate OoC data, combined with an integration of a theoretically grounded OvA classification model, UQGAN(-MCD) is able to shield each class separately from the OoC domain. This increases not only the ability to detect unknown inputs but also increases the classification accuracy on all evaluated datasets. Particularly with the integration of Monte Carlo-Dropout (MCD) into the UQGAN framework, large improvements on all datasets and evaluation metrics could be achieved. Regarding classification accuracy the UQGAN-MCD is consistently improving by $\sim 5\,\%$ over the second-best competitor and even achieves relative improvements of up to 50 % compared to closely related approaches. However, we also noticed that the generated OoC data has a negative impact on the calibration of the classifier. When measuring the ECE, UQGAN achieves low-tier results which can only be improved by applying the MCD technique. Similar results can be observed for the task of FP detection, which might actually be a result of the poor calibration. UQGAN obtains relatively low results while UQGAN-MCD achieves average results and is still surpassed by some other methods. The biggest advantage of this approach is the outstanding performance on the task of OoD detection. On all evaluated datasets, UQGAN-MCD is achieving top-1 results with often large margins of up to 65 % relative improvement over the second-best method. Both variants are surpassing related works, especially in terms of FPR95, which constitutes a significant advantage in practical applications. On the highest dimensional Tiny ImageNet 0-99 benchmark, UQGAN-MCD is even overtaking one of the oracles, which is a clear indication of better OoD generalization when utilizing OoC data for class shielding. Lastly, the ability to separate aleatoric and epistemic uncertainty gave us interesting insights into the task of adversarial example detection. We found out that there is no uncertainty type which is universally applicable for the detection of such attacks. Interestingly, both UQGAN variants are achieving superior results on the detection of untargeted PGD attacks, with absolute margins of up to 95 % for some perturbation strengths and with respect to the FPR95 evaluation metric. For the other attack types, UQGAN also achieves comparatively good results with an often increased detection rate for smaller amounts of perturbation. Another observation was that on this task, none of the two variants has an advantage over the other.

9.2 FUTURE WORK

With the increased use of DNNs in real-world applications, the area of UQ has received increased interest in the research community. As a consequence, there has been considerable progress in this field in the last years. However, as could be seen by the results in Chapter 8, there remains plenty of room for improvement in comparison to the oracle results.

Besides GANs, there are other generative models that are capable of approximating training distributions and therefore could potentially be good candidates for generating samples on the boundary of it. Especially in higher dimensional problems there remain potentially large gains if the boundary of the training distribution can be better approximated. New approaches in this area can also have a significant effect on other tasks as has been shown by the increased classification accuracy of the UQGAN-MCD approach in Chapter 8.

With respect to transferability, many UQ approaches are often not directly applicable to other machine learning tasks, e.g., object detection, semantic segmentation or regression. The task of object detection is especially difficult as these models are trained to ignore objects which are not part of the training set. Which means, besides having to identify unknown inputs to the classification part, an uncertainty measure has to cover the situation of false negatives, from which there are potentially many. Gradient metrics could very recently be transferred to the tasks of object detection [106], semantic segmentation [86] and regression in the case of depth estimation [55]. However, transferring a generative model to these tasks is not straight forward. What should a generated example for the task of object detection look like? How can we ensure proper bounding boxes for the generated objects? How can we incorporate the fact that normally an object detector is trained to ignore these inputs? Similar problems occur in the case of semantic segmentation and regression. How is an OoD or OoC example defined for a regression task? There are no categories that we can condition the model on. As most of these problems require considerable changes in the architecture and/or the theory of currently existing methods, they should be addressed in future research.

Another important point in terms of real-world applicability is the efficiency and runtime of UQ methods. For applications like, e.g., autonomous driving, it is highly critical that the uncertainty can be gathered in real-time in order to enable the car to react to unforeseen and sudden events. Additionally, available compute resources are also constrained in these systems, further restricting the space of techniques which can be applied. We observed in the experiments that approaches which require more computation time and/or hardware resources (e.g., DEs, MCD, BNNs, UQGAN-MCD) are achieving generally better results than methods that only require a single forward pass or a gradient computation (e.g., MCP, Entropy, UQGAN, EGAN). Model uncertainty is a large contributor to the overall uncertainty of an NN and quantifying it without having access to multiple model hypothesis will be very challenging. Also, many approaches are not transferable to very high dimensional tasks, e.g., high resolution images, dense prediction tasks or large models with many parameters. Future research should focus on closing the gap between sampling and deterministic approaches in order to be able to apply them in practical applications.

Until now, very few publications could give theoretical guarantees on predictions and uncertainty estimates. Especially for safety-critical applications that affect public areas, there could be demands from the legislator for guarantees about the functioning of such systems. Without proper theoretical results this task is very difficult to accomplish and future research in this area is of high interest to many practitioners.

A ADDITIONAL QUANTITATIVE RESULTS

A.1 FALSE POSITIVE DETECTION

Method	MNIST 0-4			CIFAR10 0-4		
	Accuracy ↑	AUROC S/F ↑	ECE ↓	Accuracy ↑	AUROC S/F ↑	ECE ↓
GCE	99.87 (0.02)	99.68 (0.09)	**0.11 (0.02)**	**82.42 (0.31)**	83.90 (0.82)	**11.34 (0.83)**
UQGAN	99.74 (0.05)	99.35 (0.31)	0.15 (0.05)	87.26 (0.29)	84.71 (0.52)	10.35 (0.20)
UQGAN-MCD	99.80 (0.04)	99.42 (0.11)	1.38 (0.11)	**90.26 (0.22)**	**89.19 (0.13)**	**2.34 (0.33)**
MCP	99.87 (0.02)	99.68 (0.13)	**0.11 (0.02)**	**82.42 (0.31)**	83.29 (0.89)	**11.34 (0.83)**
$\tilde{H}$	99.87 (0.02)	99.66 (0.14)	**0.11 (0.02)**	**82.42 (0.31)**	83.41 (0.88)	**11.34 (0.83)**
GKLD	99.87 (0.02)	**60.42 (6.46)**	**0.11 (0.02)**	**82.42 (0.31)**	**66.39 (2.40)**	**11.34 (0.83)**
OvA	99.84 (0.06)	**99.84 (0.06)**	0.12 (0.04)	82.82 (0.62)	82.19 (0.81)	8.62 (4.55)
MCD	99.91 (0.02)	99.62 (0.12)	0.83 (0.10)	85.08 (0.56)	83.91 (0.49)	9.90 (0.42)
BBB	**99.67 (0.02)**	99.50 (0.06)	0.78 (0.05)	84.05 (0.33)	85.22 (0.40)	9.17 (0.41)
DE	**99.89 (0.03)**	99.74 (0.08)	0.15 (0.01)	85.43 (0.22)	85.29 (0.57)	3.10 (0.29)
CCGAN	99.82 (0.02)	99.62 (0.15)	0.16 (0.01)	83.58 (0.11)	85.08 (0.18)	9.31 (0.80)
EGAN	99.70 (0.03)	98.13 (0.45)	**1.98 (0.85)**	82.46 (0.35)	82.88 (0.49)	6.71 (1.81)
$\tilde{H}$-O	99.79 (0.06)	98.93 (0.68)	0.82 (0.06)	83.41 (0.58)	80.73 (0.54)	7.53 (0.27)
OvA-O	99.77 (0.03)	99.47 (0.16)	0.14 (0.02)	83.70 (0.50)	81.27 (1.03)	8.85 (0.49)

Table A.1: Results for MNIST 0-4 as ID vs {MNIST 5-9, EMNIST-Letters, Omniglot, Fashion-MNIST, SVHN, CIFAR10} as OoD datasets. For each performance metric the best result is highlighted in bold green and the worst in bold red.

A.2 OUT-OF-DISTRIBUTION DETECTION

A.2.1 *Aggregated Results*

Method	MNIST 0-4			CIFAR10 0-4		
	AUROC ↑	AUPR-In ↑	FPR95 ↓	AUROC ↑	AUPR-In ↑	FPR95 ↓
GCE	97.03 (0.17)	69.19 (1.19)	9.44 (0.37)	71.75 (1.73)	29.91 (3.30)	**89.48 (1.60)**
UQGAN	98.03 (0.28)	80.05 (2.65)	8.73 (1.47)	86.49 (0.63)	49.08 (1.05)	45.78 (2.90)
UQGAN-MCD	**98.58 (0.25)**	**83.71 (2.40)**	**5.60 (0.77)**	**89.64 (0.23)**	**53.15 (0.27)**	**43.54 (1.56)**
MCP	97.07 (0.12)	69.00 (1.65)	9.71 (0.37)	72.52 (0.51)	30.52 (0.97)	87.68 (0.43)
$\tilde{H}$	97.13 (0.12)	68.76 (1.94)	9.65 (0.39)	72.85 (0.49)	30.43 (0.87)	85.41 (0.89)
GKLD	**78.04 (2.71)**	**11.43 (0.95)**	**43.21 (7.30)**	**71.62 (1.10)**	**18.61 (1.69)**	81.05 (0.96)
OvA	97.12 (0.17)	66.68 (1.93)	9.45 (0.56)	72.52 (2.16)	32.24 (2.14)	88.74 (1.86)
MCD	97.69 (0.16)	72.82 (2.55)	8.28 (0.39)	77.56 (1.27)	38.75 (1.40)	82.35 (1.08)
BBB	95.46 (0.26)	67.09 (2.70)	17.33 (1.06)	74.23 (0.96)	29.91 (1.61)	83.97 (1.47)
DE	97.70 (0.03)	73.09 (0.62)	7.81 (0.21)	74.24 (0.73)	32.81 (1.50)	85.07 (0.81)
CCGAN	98.15 (0.13)	78.31 (2.01)	7.65 (0.46)	73.33 (0.53)	32.32 (1.09)	85.04 (1.09)
EGAN	97.78 (0.70)	69.77 (10.20)	8.98 (1.90)	86.01 (1.60)	42.32 (2.81)	45.39 (3.26)
$\tilde{H}$-O	99.90 (0.02)	98.66 (0.28)	0.43 (0.10)	95.44 (0.28)	68.51 (0.57)	17.27 (1.44)
OvA-O	99.90 (0.01)	98.50 (0.12)	0.40 (0.03)	91.38 (0.74)	53.75 (1.49)	35.94 (3.79)

Table A.2: Results for MNIST 0-4 as ID vs. {MNIST 5-9, CIFAR10, EMNIST-Letters, Omniglot, Fashion-MNIST, SVHN} as OoD datasets and for CIFAR10 0-4 as ID vs. {CIFAR10 5-9, LSUN, SVHN, Fashion-MNIST, MNIST} as OoD datasets. For each performance metric the best result is highlighted in bold green and the worst in bold red.

A.2.2 *Results between Individual Datasets*

Method	AUROC ↑	AUPR-In ↑	FPR95 ↓	AUROC ↑	AUPR-In ↑	FPR95 ↓
	MNIST 0-4 vs. MNIST 5-9			MNIST 0-4 vs. EMNIST-Letters		
GCE	92.75 (0.89)	91.30 (1.16)	19.99 (1.43)	91.81 (0.34)	74.01 (0.96)	28.55 (0.78)
UQGAN	92.34 (2.31)	93.17 (2.78)	38.68 (7.09)	96.44 (0.76)	86.32 (2.69)	13.80 (3.06)
UQGAN-MCD	**95.50 (0.53)**	**96.22 (0.59)**	24.46 (2.20)	96.71 (0.55)	86.99 (2.49)	12.49 (1.63)
MCP	92.77 (0.55)	91.46 (1.05)	22.17 (1.04)	91.63 (0.28)	73.52 (1.02)	29.31 (0.88)
$\tilde{H}$	92.80 (0.54)	91.20 (1.27)	22.11 (1.04)	91.68 (0.27)	73.51 (1.14)	29.23 (0.91)
GKLD	**39.16 (0.94)**	**41.88 (0.39)**	**87.51 (2.38)**	**48.08 (2.28)**	**17.48 (0.79)**	**86.33 (2.73)**
OvA	93.65 (0.97)	92.32 (1.36)	20.06 (2.98)	91.32 (0.31)	70.45 (1.43)	29.30 (1.12)
MCD	94.32 (1.10)	93.05 (1.79)	17.27 (1.60)	92.80 (0.37)	76.85 (1.53)	26.72 (1.00)
BBB	93.73 (0.99)	92.74 (1.58)	22.17 (1.40)	90.59 (0.80)	71.60 (2.22)	34.28 (1.54)
DE	94.20 (0.24)	92.82 (0.18)	**16.89 (1.09)**	93.08 (0.10)	77.32 (0.73)	24.65 (0.37)
CCGAN	95.33 (0.74)	95.43 (1.00)	19.74 (1.46)	94.60 (0.41)	81.71 (1.56)	22.28 (1.87)
EGAN	88.91 (2.64)	86.09 (4.55)	40.49 (5.53)	**99.27 (0.30)**	**97.05 (1.22)**	**3.61 (1.48)**
$\tilde{H}$-O	99.76 (0.04)	99.77 (0.03)	0.93 (0.12)	99.84 (0.04)	99.43 (0.12)	0.73 (0.21)
OvA-O	99.65 (0.04)	99.66 (0.03)	1.21 (0.13)	99.87 (0.01)	99.49 (0.05)	0.59 (0.10)
	MNIST 0-4 vs. Omniglot			MNIST 0-4 vs. Fashion-MNIST		
GCE	98.29 (0.25)	96.54 (0.59)	6.20 (0.89)	99.30 (0.13)	99.14 (0.19)	1.16 (0.65)
UQGAN	96.70 (0.52)	94.25 (1.07)	18.00 (3.26)	99.36 (0.19)	99.04 (0.32)	2.19 (1.36)
UQGAN-MCD	98.45 (0.24)	96.94 (0.37)	5.95 (1.08)	99.65 (0.26)	99.52 (0.35)	0.68 (0.98)
MCP	98.54 (0.06)	97.10 (0.24)	5.52 (0.50)	99.29 (0.23)	99.03 (0.36)	1.79 (1.34)
$\tilde{H}$	98.59 (0.06)	97.15 (0.24)	5.36 (0.50)	99.35 (0.22)	99.06 (0.36)	1.74 (1.31)
GKLD	**79.02 (3.05)**	**57.04 (4.14)**	**57.40 (5.72)**	**90.02 (5.66)**	**79.32 (0.49)**	**29.63 (13.42)**
OvA	98.75 (0.11)	97.60 (0.22)	4.75 (0.74)	99.39 (0.12)	99.15 (0.16)	1.60 (0.53)
MCD	**98.96 (0.09)**	97.56 (0.27)	3.85 (0.43)	99.69 (0.05)	99.52 (0.08)	0.75 (0.21)
BBB	96.82 (0.16)	95.22 (0.30)	14.86 (0.56)	97.93 (0.18)	97.64 (0.24)	8.02 (1.30)
DE	98.93 (0.04)	**97.77 (0.07)**	**3.76 (0.20)**	99.59 (0.09)	99.38 (0.15)	1.01 (0.63)
CCGAN	98.35 (0.29)	96.22 (0.83)	6.78 (1.06)	**99.95 (0.01)**	**99.91 (0.03)**	**0.06 (0.02)**
EGAN	91.03 (3.53)	78.34 (9.87)	36.96 (9.71)	99.92 (0.02)	99.84 (0.04)	0.23 (0.11)
$\tilde{H}$-O	99.68 (0.09)	99.25 (0.21)	1.26 (0.35)	100.00 (0.00)	100.00 (0.00)	0.00 (0.00)
OvA-O	99.68 (0.05)	99.18 (0.13)	1.22 (0.17)	100.00 (0.00)	100.00 (0.00)	0.00 (0.00)
	MNIST 0-4 vs. SVHN			MNIST 0-4 vs. CIFAR10		
GCE	99.54 (0.09)	99.02 (0.27)	0.43 (0.20)	99.47 (0.10)	99.37 (0.15)	0.56 (0.29)
UQGAN	99.83 (0.09)	99.50 (0.21)	0.17 (0.06)	99.84 (0.08)	99.76 (0.11)	0.21 (0.12)
UQGAN-MCD	99.80 (0.22)	99.47 (0.53)	0.32 (0.40)	99.84 (0.19)	99.78 (0.26)	0.28 (0.47)
MCP	99.68 (0.12)	99.14 (0.24)	0.39 (0.12)	99.54 (0.13)	99.39 (0.17)	0.59 (0.30)
$\tilde{H}$	99.75 (0.11)	99.22 (0.23)	0.37 (0.11)	99.61 (0.13)	99.45 (0.17)	0.56 (0.29)
GKLD	97.43 (2.68)	**86.72 (11.38)**	8.79 (8.98)	**95.49 (4.38)**	**90.98 (8.31)**	**16.45 (14.50)**
OvA	99.76 (0.07)	99.30 (0.17)	0.36 (0.12)	99.55 (0.16)	99.39 (0.21)	0.75 (0.45)
MCD	99.94 (0.01)	99.75 (0.05)	0.15 (0.03)	99.92 (0.03)	99.87 (0.05)	0.09 (0.05)
BBB	**97.20 (0.32)**	95.75 (0.34)	**11.46 (3.55)**	97.52 (0.40)	97.48 (0.35)	7.44 (2.13)
DE	99.89 (0.01)	99.55 (0.05)	0.25 (0.06)	99.82 (0.05)	99.73 (0.07)	0.20 (0.09)
CCGAN	**100.00 (0.00)**	**100.00 (0.00)**	**0.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.00)**	**0.00 (0.00)**
EGAN	**100.00 (0.00)**	**100.00 (0.00)**	**0.00 (0.00)**	**100.00 (0.00)**	**100.00 (0.01)**	0.01 (0.01)
$\tilde{H}$-O	100.00 (0.00)	100.00 (0.00)	0.00 (0.00)	100.00 (0.00)	100.00 (0.00)	0.00 (0.00)
OvA-O	100.00 (0.00)	100.00 (0.00)	0.00 (0.00)	100.00 (0.00)	100.00 (0.00)	0.00 (0.00)

Table A.3: An OoD dataset-wise breakdown of the MNIST 0-4 results given in Table A.2. In each column of each dataset comparison, the best value is marked in bold green and the worst one in bold red.

Method	AUROC ↑	AUPR-In ↑	FPR95 ↓	AUROC ↑	AUPR-In ↑	FPR95 ↓
	CIFAR10 0-4 vs. CIFAR10 5-9			CIFAR10 0-4 vs. LSUN		
GCE	65.25 (0.90)	67.09 (1.01)	91.32 (0.87)	72.98 (1.04)	64.64 (1.26)	**88.50 (1.44)**
UQGAN	65.94 (0.29)	**72.26 (0.30)**	86.94 (1.12)	76.45 (0.52)	69.35 (0.56)	80.50 (1.31)
UQGAN-MCD	**71.31 (0.49)**	71.98 (0.35)	**84.07 (0.76)**	**82.52 (0.72)**	**76.03 (0.80)**	**72.93 (1.14)**
MCP	64.45 (0.55)	65.94 (0.80)	90.73 (0.33)	72.84 (0.59)	63.69 (0.93)	87.11 (0.56)
$\tilde{H}$	64.64 (0.53)	65.82 (0.69)	89.50 (0.69)	73.33 (0.51)	63.95 (0.90)	83.58 (0.56)
GKLD	**54.55 (1.51)**	**53.84 (1.00)**	**91.51 (0.79)**	**61.68 (2.18)**	**43.81 (2.63)**	86.42 (0.73)
OvA	65.31 (0.67)	66.28 (0.76)	88.15 (0.52)	74.94 (0.79)	66.33 (1.06)	82.50 (1.03)
MCD	63.52 (0.33)	64.11 (0.38)	90.01 (0.39)	77.04 (0.22)	70.27 (0.41)	81.53 (0.59)
BBB	66.78 (0.34)	67.16 (1.07)	88.50 (0.62)	75.31 (0.65)	66.79 (1.13)	82.64 (1.16)
DE	66.85 (0.38)	67.64 (0.57)	87.59 (0.42)	78.03 (0.24)	69.56 (0.47)	77.07 (0.66)
CCGAN	65.58 (0.13)	66.94 (0.18)	88.46 (0.64)	75.25 (0.33)	67.26 (0.41)	81.66 (0.99)
EGAN	65.56 (0.50)	66.67 (0.51)	89.09 (1.17)	75.82 (1.22)	67.74 (1.65)	83.19 (1.91)
$\tilde{H}$-O	73.21 (0.52)	75.00 (0.41)	81.41 (0.88)	98.27 (0.09)	96.81 (0.14)	7.91 (0.45)
OvA-O	69.45 (0.73)	69.58 (0.66)	84.92 (0.92)	95.92 (0.38)	92.38 (0.65)	19.36 (2.03)
	CIFAR10 0-4 vs. SVHN			CIFAR10 0-4 vs. Fashion-MNIST		
GCE	70.64 (2.80)	42.29 (5.25)	85.95 (1.89)	**72.16 (1.72)**	**65.53 (1.83)**	**92.24 (1.97)**
UQGAN	98.50 (0.26)	92.05 (1.07)	5.99 (1.27)	78.78 (0.88)	72.41 (0.85)	81.09 (2.84)
UQGAN-MCD	**98.93 (0.12)**	**94.40 (0.62)**	**5.09 (0.44)**	**84.75 (1.32)**	**80.48 (1.45)**	**76.11 (5.07)**
MCP	70.96 (1.87)	44.63 (2.22)	88.76 (1.38)	73.90 (1.18)	67.14 (1.35)	88.48 (0.82)
$\tilde{H}$	71.23 (1.94)	44.81 (2.17)	87.04 (2.22)	73.93 (1.26)	67.18 (1.38)	88.47 (1.71)
GKLD	**69.03 (1.41)**	**34.75 (3.38)**	85.19 (1.75)	81.11 (1.47)	74.12 (2.02)	76.69 (2.73)
OvA	70.07 (4.23)	45.76 (4.99)	**91.83 (3.14)**	73.94 (1.31)	68.28 (1.45)	89.25 (1.33)
MCD	76.73 (2.77)	58.97 (4.10)	84.98 (1.87)	81.85 (0.75)	77.62 (0.79)	80.75 (1.85)
BBB	76.21 (0.58)	50.10 (1.76)	80.85 (1.42)	74.51 (1.34)	68.52 (1.75)	88.11 (1.87)
DE	72.02 (1.11)	45.95 (2.24)	88.13 (0.61)	72.82 (1.20)	66.38 (2.02)	89.97 (0.84)
CCGAN	73.60 (0.61)	48.68 (1.08)	85.45 (1.08)	74.47 (0.52)	68.40 (0.77)	87.47 (0.60)
EGAN	98.43 (0.42)	91.36 (1.82)	5.62 (1.68)	72.44 (3.94)	66.54 (4.86)	89.38 (2.57)
$\tilde{H}$-O	96.85 (0.62)	88.28 (1.42)	14.26 (2.88)	97.89 (0.16)	96.65 (0.28)	9.50 (0.61)
OvA-O	89.47 (1.47)	70.44 (2.54)	48.76 (7.42)	96.63 (0.40)	94.34 (0.71)	17.37 (2.48)
	CIFAR10 0-4 vs. MNIST					
GCE	76.23 (0.64)	73.21 (0.58)	**95.87 (1.34)**			
UQGAN	83.24 (3.79)	75.49 (4.07)	58.75 (13.88)			
UQGAN-MCD	86.64 (1.18)	**82.03 (1.28)**	61.32 (5.75)			
MCP	78.89 (1.79)	74.29 (2.40)	83.07 (2.10)			
$\tilde{H}$	79.59 (1.81)	74.66 (2.38)	77.57 (3.12)			
GKLD	87.37 (1.61)	84.61 (2.51)	64.02 (2.07)			
OvA	78.65 (0.75)	75.69 (0.57)	86.70 (2.27)			
MCD	82.98 (1.21)	79.05 (1.68)	73.99 (1.97)			
BBB	**71.46 (4.50)**	**62.19 (6.23)**	87.01 (3.35)			
DE	81.33 (1.54)	76.97 (1.67)	78.85 (3.47)			
CCGAN	73.41 (2.43)	64.90 (4.16)	83.21 (2.58)			
EGAN	**87.63 (3.97)**	80.69 (5.53)	**45.16 (11.84)**			
$\tilde{H}$-O	97.59 (0.26)	97.10 (0.30)	10.16 (2.03)			
OvA-O	97.56 (0.20)	96.52 (0.40)	13.25 (1.95)			

Table A.4: An OoD dataset-wise breakdown of the CIFAR10 0-4 results given in Table A.2. In each column of each dataset comparison, the best value is marked in bold green and the worst one in bold red.

Method	AUROC ↑	AUPR-In ↑	FPR95 ↓	AUROC ↑	AUPR-In ↑	FPR95 ↓
	CIFAR100 0-49 vs. CIFAR100 50-99			CIFAR100 0-49 vs. LSUN		
GCE	**50.00 (0.00)**	50.10 (0.03)	**94.98 (0.00)**	63.75 (0.38)	52.72 (0.39)	92.68 (0.49)
UQGAN	64.52 (0.17)	65.00 (0.35)	89.57 (0.47)	65.30 (0.56)	52.94 (0.30)	90.36 (0.61)
UQGAN-MCD	**66.97 (0.30)**	65.14 (0.37)	**87.74 (0.48)**	68.60 (0.62)	56.80 (0.81)	88.77 (0.65)
MCP	62.43 (0.72)	62.87 (1.41)	90.92 (0.48)	65.21 (1.34)	53.47 (1.45)	89.96 (1.00)
$\tilde{H}$	63.53 (0.62)	63.45 (1.36)	90.23 (0.76)	66.62 (1.51)	54.37 (1.65)	88.76 (1.16)
GKLD	**50.00 (0.00)**	**50.01 (0.00)**	**94.99 (0.01)**	**60.62 (0.39)**	**42.17 (0.83)**	90.51 (0.52)
OvA	61.62 (0.31)	61.30 (0.42)	91.49 (0.37)	64.09 (1.09)	51.10 (1.14)	90.81 (0.92)
MCD	62.97 (0.22)	62.21 (0.41)	90.34 (0.37)	67.19 (1.00)	56.12 (0.89)	87.38 (1.22)
BBB	64.16 (0.36)	64.56 (0.61)	90.44 (0.47)	67.02 (0.84)	55.60 (1.09)	90.05 (0.57)
DE	66.95 (0.20)	**65.94 (0.39)**	87.90 (0.47)	**71.34 (0.64)**	**59.76 (0.85)**	**84.60 (1.38)**
CCGAN	62.39 (0.67)	62.31 (0.12)	90.10 (0.54)	64.24 (0.83)	51.61 (0.74)	89.53 (0.94)
EGAN	62.66 (0.40)	61.82 (0.50)	89.89 (0.69)	62.59 (1.01)	51.97 (0.51)	**93.96 (1.29)**
$\tilde{H}$-O	64.42 (0.31)	65.50 (0.31)	89.87 (0.86)	70.00 (0.33)	58.01 (0.24)	85.32 (0.88)
OvA-O	67.55 (1.07)	66.49 (1.24)	86.70 (0.53)	78.10 (0.99)	64.04 (1.12)	70.25 (2.55)
	CIFAR100 0-49 vs. SVHN			CIFAR100 0-49 vs. Fashion-MNIST		
GCE	70.97 (1.28)	41.46 (2.44)	80.27 (1.78)	61.01 (1.61)	53.51 (1.76)	**95.26 (1.90)**
UQGAN	**96.47 (1.26)**	**81.59 (5.55)**	**11.95 (3.93)**	62.98 (1.71)	51.89 (2.88)	92.37 (1.04)
UQGAN-MCD	95.63 (1.27)	80.03 (4.04)	15.47 (4.21)	64.68 (2.02)	57.46 (1.99)	91.50 (1.15)
MCP	66.32 (2.15)	37.61 (3.13)	89.73 (1.60)	68.46 (0.56)	60.24 (0.85)	89.36 (0.46)
$\tilde{H}$	68.09 (2.54)	38.90 (3.51)	89.65 (1.44)	68.95 (0.48)	60.59 (0.68)	87.66 (0.75)
GKLD	66.38 (2.56)	28.83 (4.07)	88.44 (2.42)	78.17 (1.17)	72.80 (1.32)	82.50 (2.15)
OvA	**60.63 (3.90)**	**26.37 (6.04)**	**93.11 (1.92)**	59.80 (2.67)	51.35 (3.14)	95.13 (0.65)
MCD	65.65 (2.51)	35.78 (3.03)	91.07 (1.95)	**70.50 (1.49)**	**63.75 (1.52)**	88.43 (2.58)
BBB	72.62 (1.07)	42.79 (1.62)	81.59 (2.61)	66.22 (2.02)	57.55 (3.02)	91.91 (1.20)
DE	75.01 (1.06)	49.77 (2.19)	87.10 (1.26)	67.42 (1.14)	60.17 (1.35)	**85.46 (1.12)**
CCGAN	68.73 (0.49)	39.41 (0.95)	87.73 (0.71)	67.04 (0.89)	56.88 (0.96)	86.27 (1.46)
EGAN	92.83 (2.72)	71.54 (8.20)	23.57 (7.35)	**59.12 (6.18)**	**49.44 (5.14)**	94.58 (3.06)
$\tilde{H}$-O	86.97 (0.75)	63.53 (1.22)	50.09 (2.55)	97.47 (0.45)	94.63 (0.85)	10.94 (1.96)
OvA-O	98.17 (0.22)	90.82 (0.89)	8.11 (0.98)	99.62 (0.05)	99.13 (0.12)	1.65 (0.23)
	CIFAR100 0-49 vs. MNIST					
GCE	**62.85 (1.49)**	**60.16 (2.08)**	**98.71 (1.45)**			
UQGAN	77.28 (5.81)	70.28 (6.06)	78.32 (10.93)			
UQGAN-MCD	77.13 (3.87)	73.64 (4.41)	90.07 (2.55)			
MCP	75.57 (4.00)	68.72 (4.65)	81.22 (5.60)			
$\tilde{H}$	79.14 (4.38)	72.10 (5.53)	76.56 (8.39)			
GKLD	89.52 (1.53)	86.48 (1.87)	54.69 (5.59)			
OvA	71.94 (3.24)	64.45 (3.29)	90.07 (2.67)			
MCD	73.41 (1.90)	66.48 (3.12)	87.39 (2.49)			
BBB	71.27 (2.11)	63.84 (3.33)	91.43 (1.92)			
DE	**85.92 (1.62)**	**82.02 (1.88)**	**68.03 (5.45)**			
CCGAN	77.66 (1.41)	69.86 (1.33)	76.68 (1.55)			
EGAN	77.88 (6.84)	71.93 (7.67)	80.26 (14.55)			
$\tilde{H}$-O	99.79 (0.06)	99.56 (0.12)	1.02 (0.28)			
OvA-O	99.99 (0.01)	99.97 (0.01)	0.03 (0.03)			

Table A.5: An OoD dataset-wise breakdown of the CIFAR100 0-49 results given in Table 8.5. In each column of each dataset comparison, the best value is marked in bold green and the worst one in bold red.

Method	AUROC ↑	AUPR-In ↑	FPR95 ↓	AUROC ↑	AUPR-In ↑	FPR95 ↓
	Tiny ImageNet 0-99 vs. Tiny ImageNet 100-199			Tiny ImageNet 0-99 vs. SVHN		
GCE	57.51 (0.40)	60.56 (0.60)	93.66 (0.50)	76.14 (2.28)	42.26 (2.33)	72.46 (5.69)
UQGAN	59.16 (0.52)	61.14 (0.44)	93.10 (0.35)	98.98 (0.44)	93.46 (2.87)	3.49 (1.31)
UQGAN-MCD	**62.01 (0.22)**	**64.70 (0.27)**	**91.97 (0.40)**	**99.39 (0.41)**	**96.18 (2.67)**	**2.19 (1.36)**
MCP	58.16 (0.25)	60.98 (0.30)	93.22 (0.52)	63.33 (1.39)	32.44 (2.05)	90.93 (1.05)
$\tilde{H}$	58.69 (0.30)	61.42 (0.31)	93.30 (0.27)	65.35 (1.48)	34.25 (1.80)	91.90 (1.66)
GKLD	57.44 (0.44)	**57.92 (0.62)**	93.05 (0.72)	62.91 (2.57)	**30.05 (4.20)**	95.10 (2.05)
OvA	58.53 (0.45)	60.28 (0.33)	93.29 (0.27)	**59.65 (4.57)**	32.66 (4.11)	**97.02 (0.95)**
MCD	61.12 (0.40)	64.16 (0.39)	93.11 (0.62)	63.54 (4.81)	37.05 (3.30)	95.33 (1.73)
BBB	57.99 (0.47)	60.44 (0.42)	93.18 (0.60)	74.18 (1.78)	38.32 (1.70)	73.54 (5.25)
DE	60.70 (0.23)	64.15 (0.23)	93.16 (0.34)	73.51 (0.81)	48.67 (1.31)	90.76 (1.18)
CCGAN	58.45 (0.23)	61.26 (0.31)	93.22 (0.53)	61.86 (1.66)	31.25 (1.56)	92.79 (0.76)
EGAN	**56.32 (0.80)**	58.82 (0.48)	**94.07 (0.60)**	91.40 (6.55)	71.24 (16.70)	24.80 (15.25)
$\tilde{H}$-O	58.57 (0.73)	61.46 (0.75)	93.23 (0.68)	84.04 (2.72)	57.38 (4.13)	57.04 (8.27)
OvA-O	60.16 (0.31)	61.23 (0.23)	91.92 (0.57)	99.39 (0.19)	95.80 (1.31)	2.25 (0.70)
	Tiny ImageNet 0-99 vs. Fashion-MNIST			Tiny ImageNet 0-99 vs. MNIST		
GCE	58.74 (5.77)	47.83 (4.52)	94.31 (3.37)	**50.16 (4.73)**	**43.98 (3.78)**	**98.11 (2.08)**
UQGAN	55.53 (4.69)	42.81 (4.04)	93.93 (3.42)	61.69 (3.34)	49.40 (3.00)	91.39 (3.62)
UQGAN-MCD	**95.06 (1.45)**	**89.18 (2.88)**	**17.29 (4.71)**	**99.78 (0.18)**	**99.52 (0.39)**	**1.04 (0.94)**
MCP	61.16 (2.40)	51.69 (2.62)	94.55 (1.42)	58.91 (2.05)	48.66 (2.13)	94.19 (1.70)
$\tilde{H}$	60.40 (2.88)	52.07 (2.77)	97.10 (1.16)	58.77 (2.97)	49.18 (2.26)	95.64 (2.23)
GKLD	65.97 (4.36)	59.63 (3.65)	94.10 (2.68)	78.40 (2.26)	72.64 (2.37)	78.92 (4.49)
OvA	**45.63 (4.00)**	**41.60 (3.30)**	**99.70 (0.21)**	51.48 (10.97)	45.52 (10.95)	97.71 (2.51)
MCD	56.34 (6.24)	53.15 (4.84)	98.79 (0.84)	71.00 (3.85)	66.16 (4.66)	95.63 (2.55)
BBB	61.83 (2.86)	50.29 (2.99)	91.78 (2.26)	63.37 (5.57)	48.91 (6.19)	87.10 (5.19)
DE	58.77 (1.34)	55.05 (2.29)	99.35 (0.25)	65.32 (1.95)	59.31 (1.74)	96.40 (1.83)
CCGAN	58.59 (2.13)	49.75 (1.96)	97.04 (0.84)	57.31 (5.67)	49.00 (5.73)	96.97 (1.45)
EGAN	72.10 (11.72)	60.00 (14.00)	67.91 (17.06)	93.78 (4.14)	87.74 (6.11)	22.91 (18.67)
$\tilde{H}$-O	91.63 (2.01)	85.61 (3.03)	35.67 (8.17)	96.29 (1.14)	93.32 (2.04)	17.98 (5.19)
OvA-O	99.81 (0.11)	99.50 (0.27)	0.74 (0.43)	100 (0.00)	100 (0.00)	0.00 (0.00)

Table A.6: An OoD dataset-wise breakdown of the Tiny ImageNet 0-99 results given in Table 8.5. In each column of each dataset comparison, the best value is marked in bold green and the worst one in bold red.

A.3 ADVERSARIAL EXAMPLE DETECTION

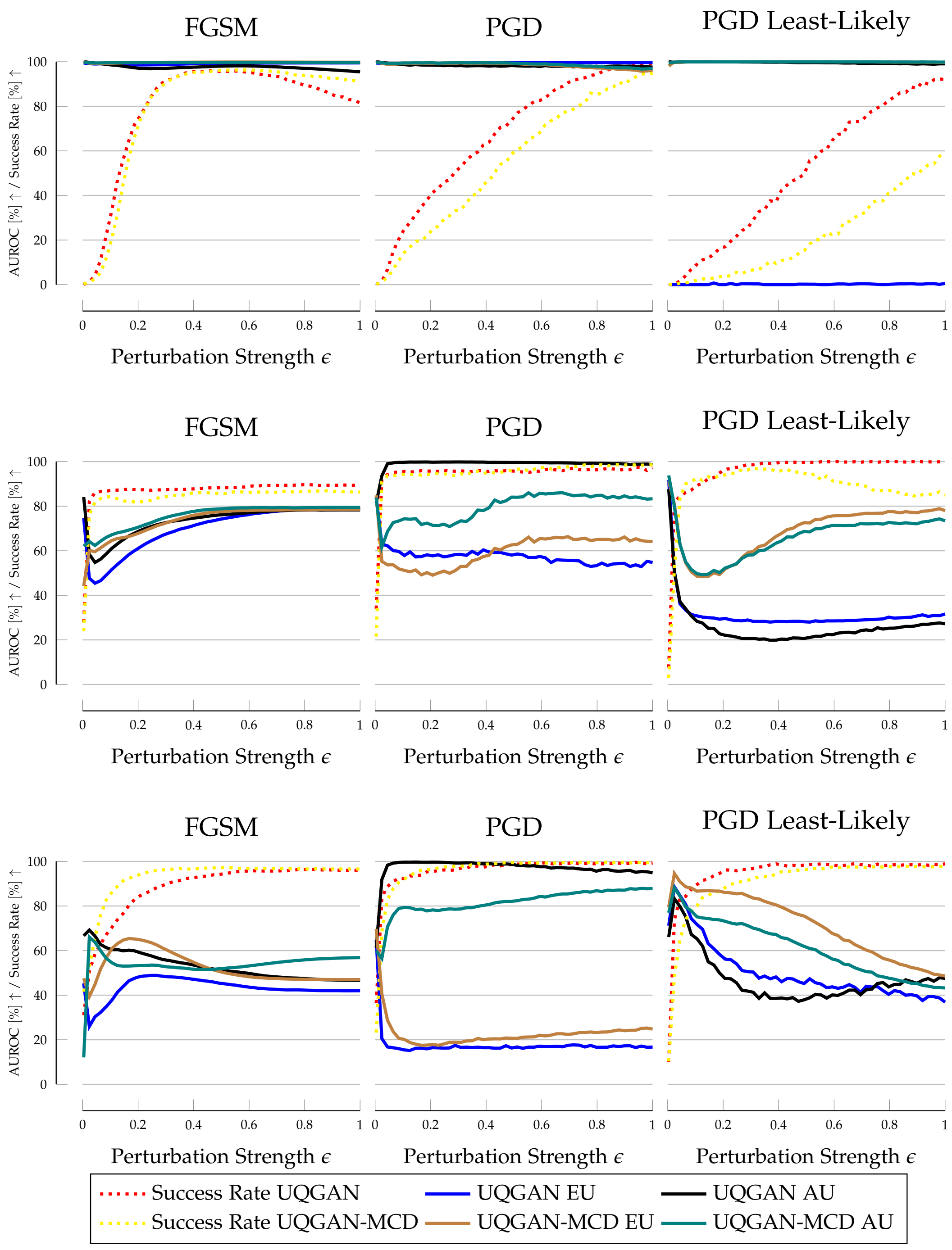

Figure A.1: Success and detection rates on the UQGAN and UQGAN-MCD models trained on the MNIST 0-4 (top row), CIFAR10 0-4 (middle row) and CIFAR100 0-49 (bottom row) datasets. Adversarial examples are generated by the FGSM [43] (left column), PGD [88] (middle column) and PGD Least-Likely (right column) with a white-box attack. For each model the aleatoric as well as the epistemic uncertainties are considered for the task of adversarial example detection measured with the AUROC

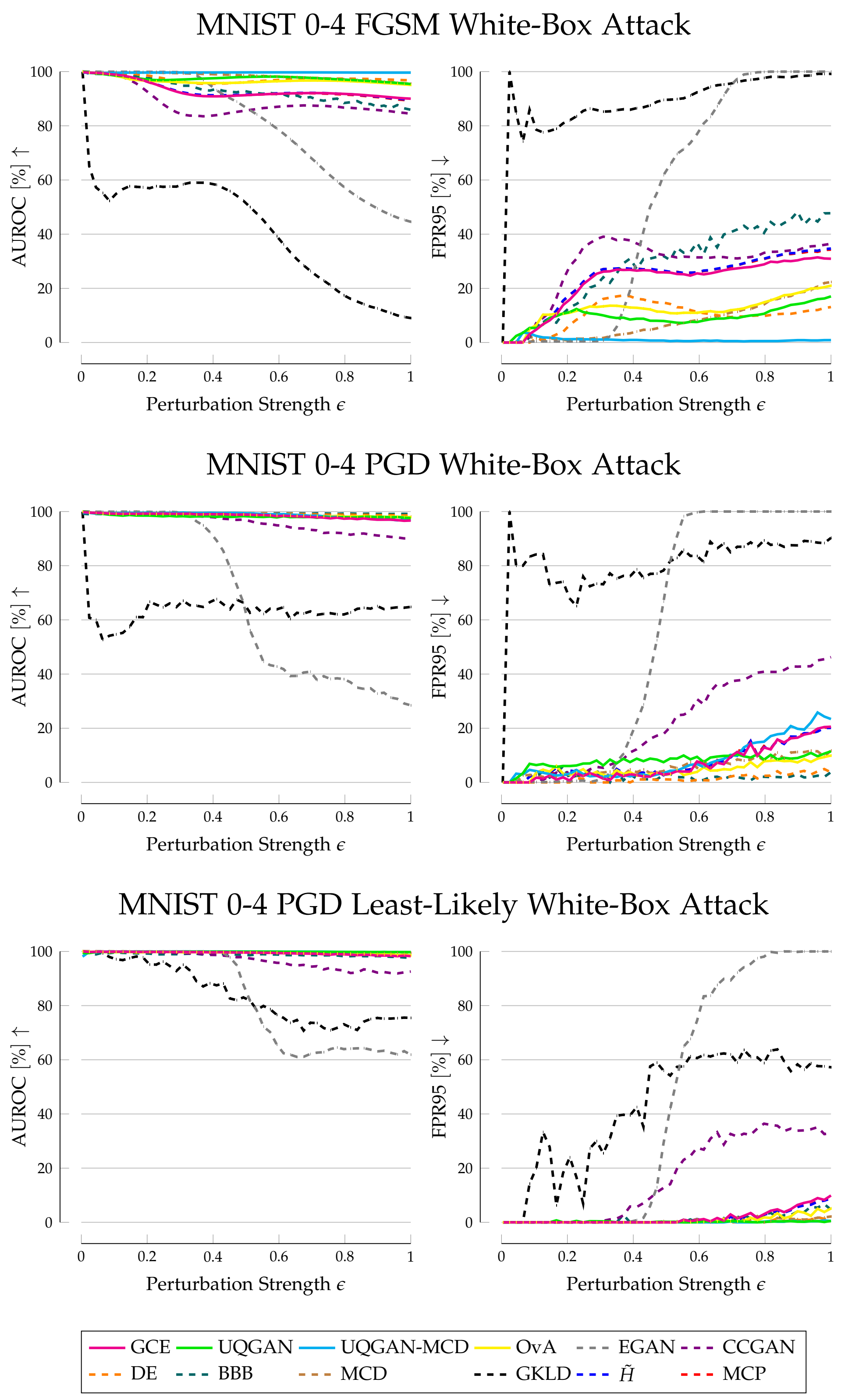

Figure A.2: Detection rates of adversarial examples generated by the FGSM [43] (top), PGD [88] (middle) and PGD Least-Likely (bottom) with a white-box attack on the MNIST 0-4 dataset. Evaluation metrics are AUROC (left) and FPR95 (right) and are computed on the aleatoric uncertainty for the non-targeted attacks (FGSM and PGD) and on the epistemic uncertainty for the targeted PGD attack. Methods from this work are marked with a solid line.

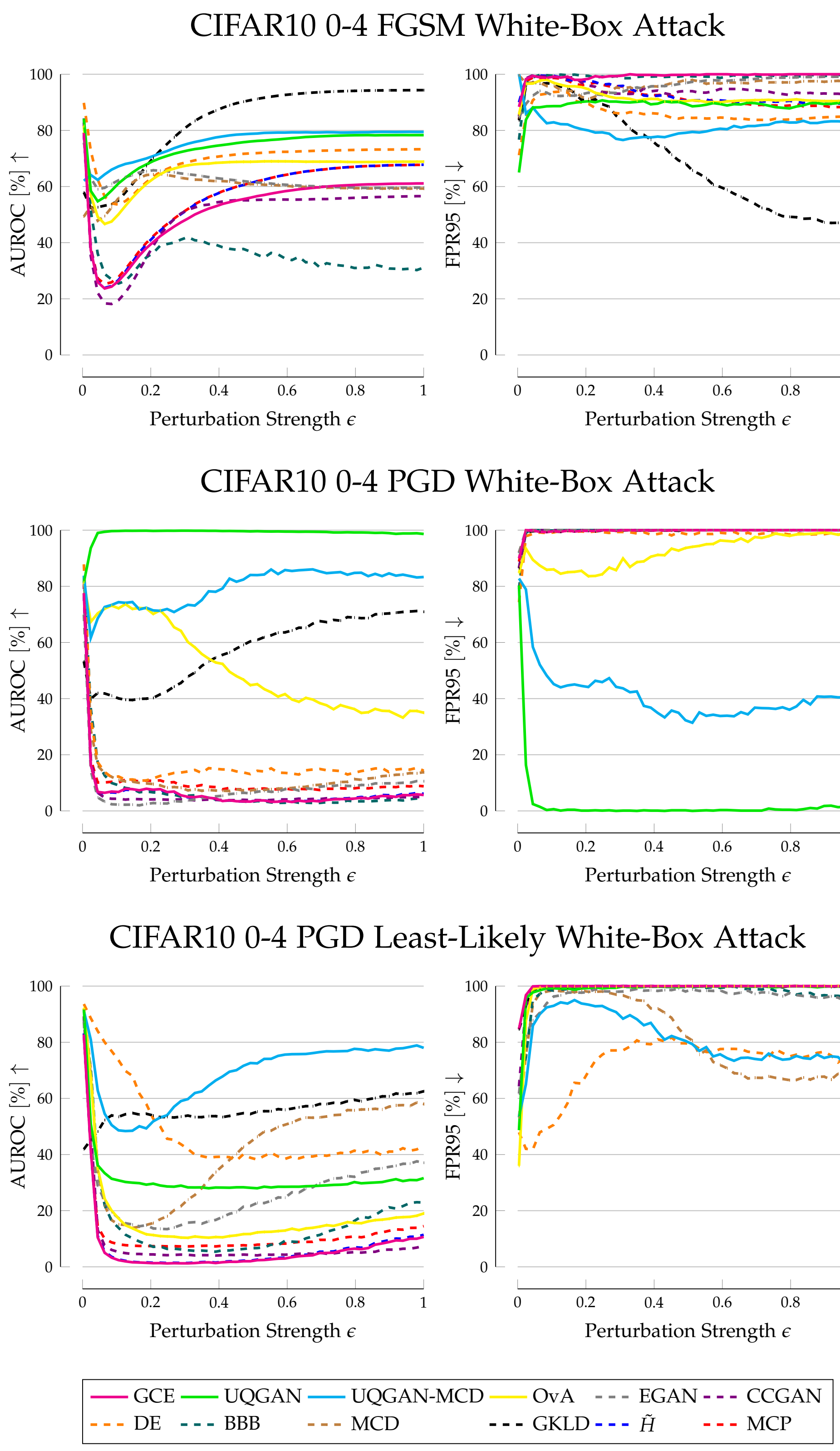

Figure A.3: Detection rates of adversarial examples generated by the FGSM [43] (top), PGD [88] (middle) and PGD Least-Likely (bottom) with a white-box attack on the CIFAR10 0-4 dataset. Evaluation metrics are AUROC (left) and FPR95 (right) and are computed on the aleatoric uncertainty for the non-targeted attacks (FGSM and PGD) and on the epistemic uncertainty for the targeted PGD attack. Methods from this work are marked with a solid line.

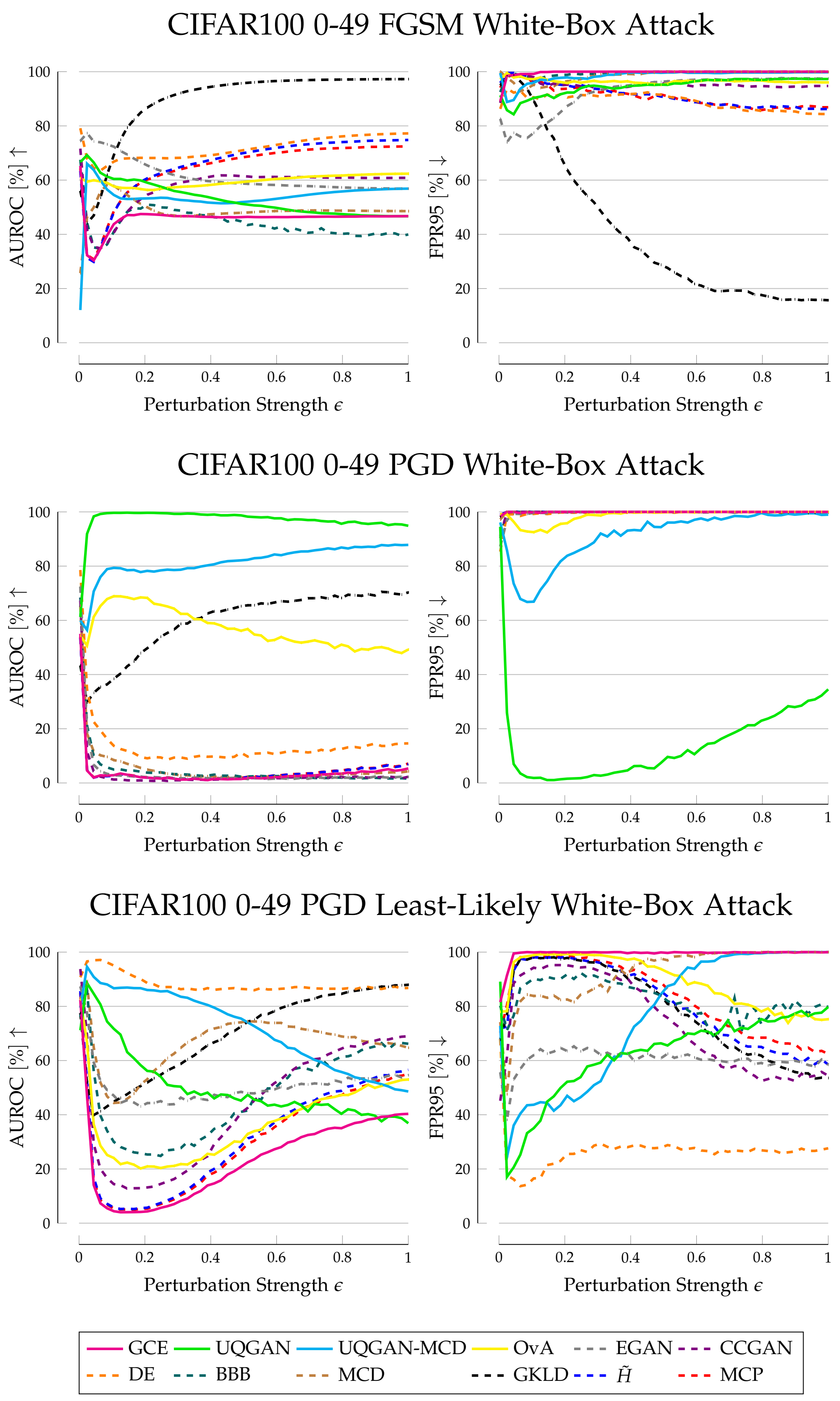

Figure A.4: Detection rates of adversarial examples generated by the FGSM [43] (top), PGD [88] (middle) and PGD Least-Likely (bottom) with a white-box attack on the CIFAR100 0-49 dataset. Evaluation metrics are AUROC (left) and FPR95 (right) and are computed on the aleatoric uncertainty for the non-targeted attacks (FGSM and PGD) and on the epistemic uncertainty for the targeted PGD attack. Methods from this work are marked with a solid line.

B

HYPERPARAMETER-STUDY RESULTS

B.1 GRADIENT METRICS AND LAYER CHOICE

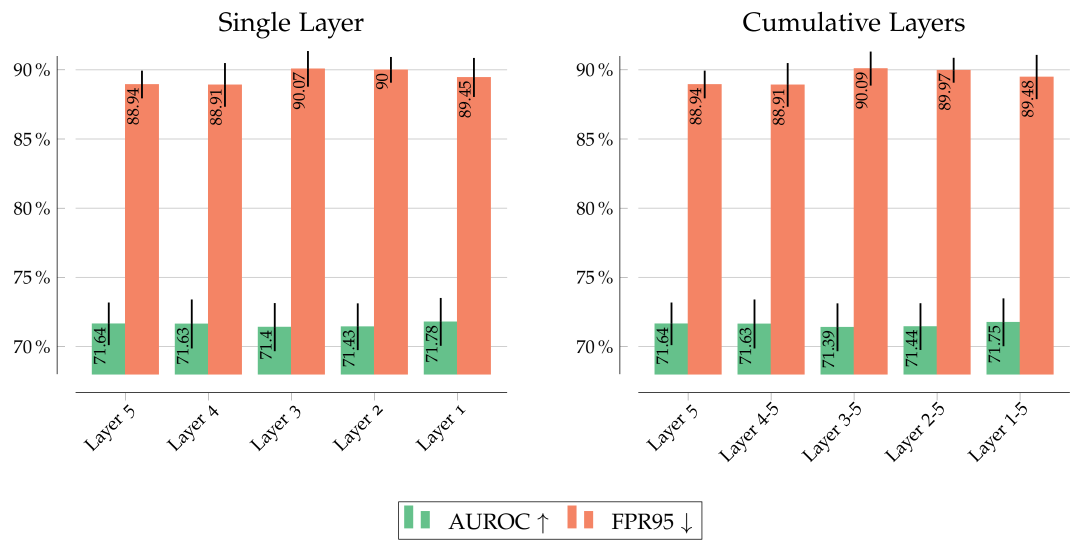

Figure B.1: AUROC and FPR95 for the min gradient metric on different layers. Left: Gradient metric from individual layers, with layer 5 being the closest to the output and layer 1 being the closest to the input of the NN. Right: Gradient metric for multiple layers, starting from the closest to the output, successively adding more layers. Layer 1-5 is the same as over the whole network.

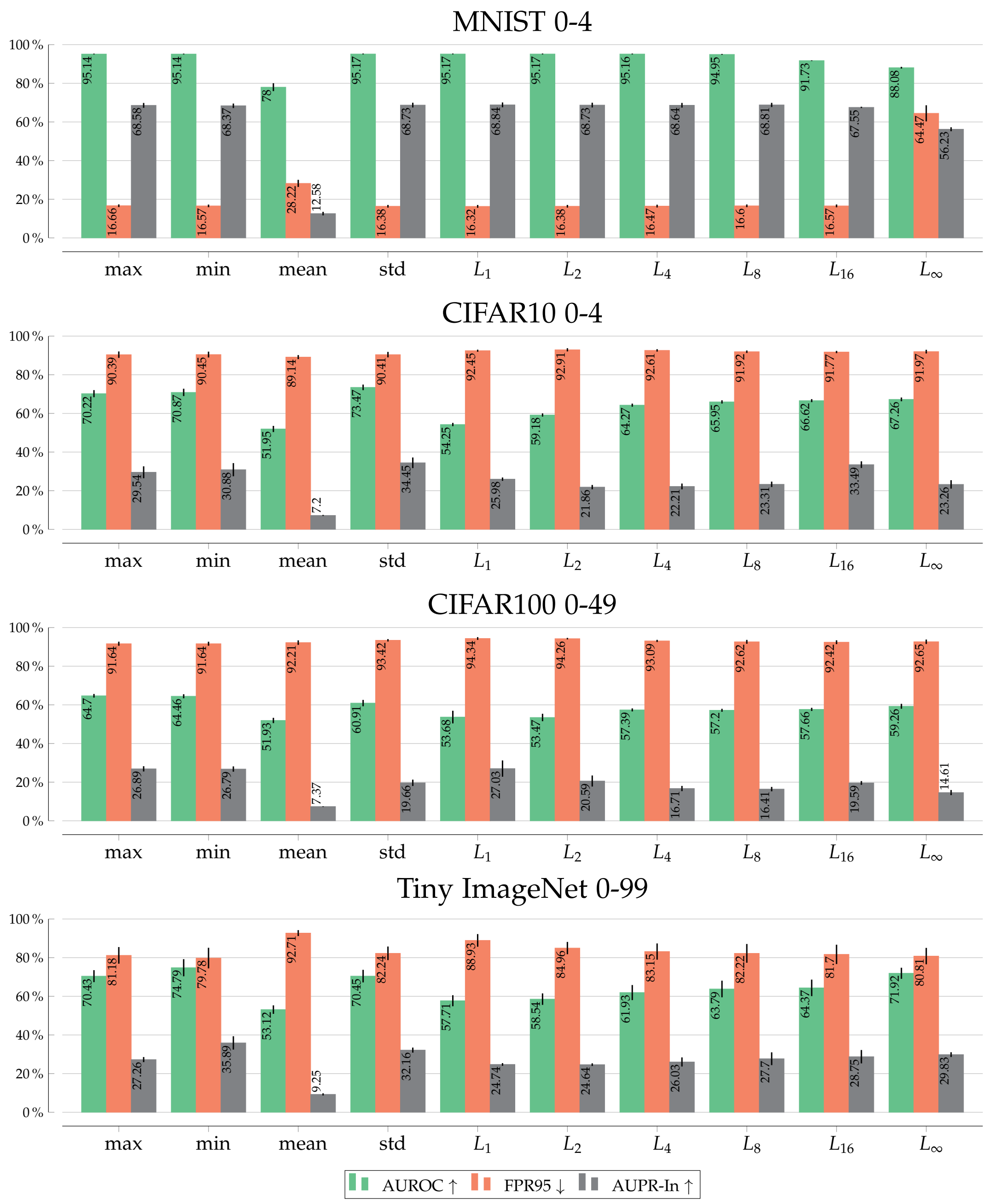

Figure B.2: AUROC, FPR95 and AUPR-In for 10 different gradient metrics on the MNIST 0-4, CIFAR10 0-4, CIFAR100 0-49 and Tiny ImageNet 0-99 evaluation sets.

B.2 UQGAN HYPERPARAMETER STUDY

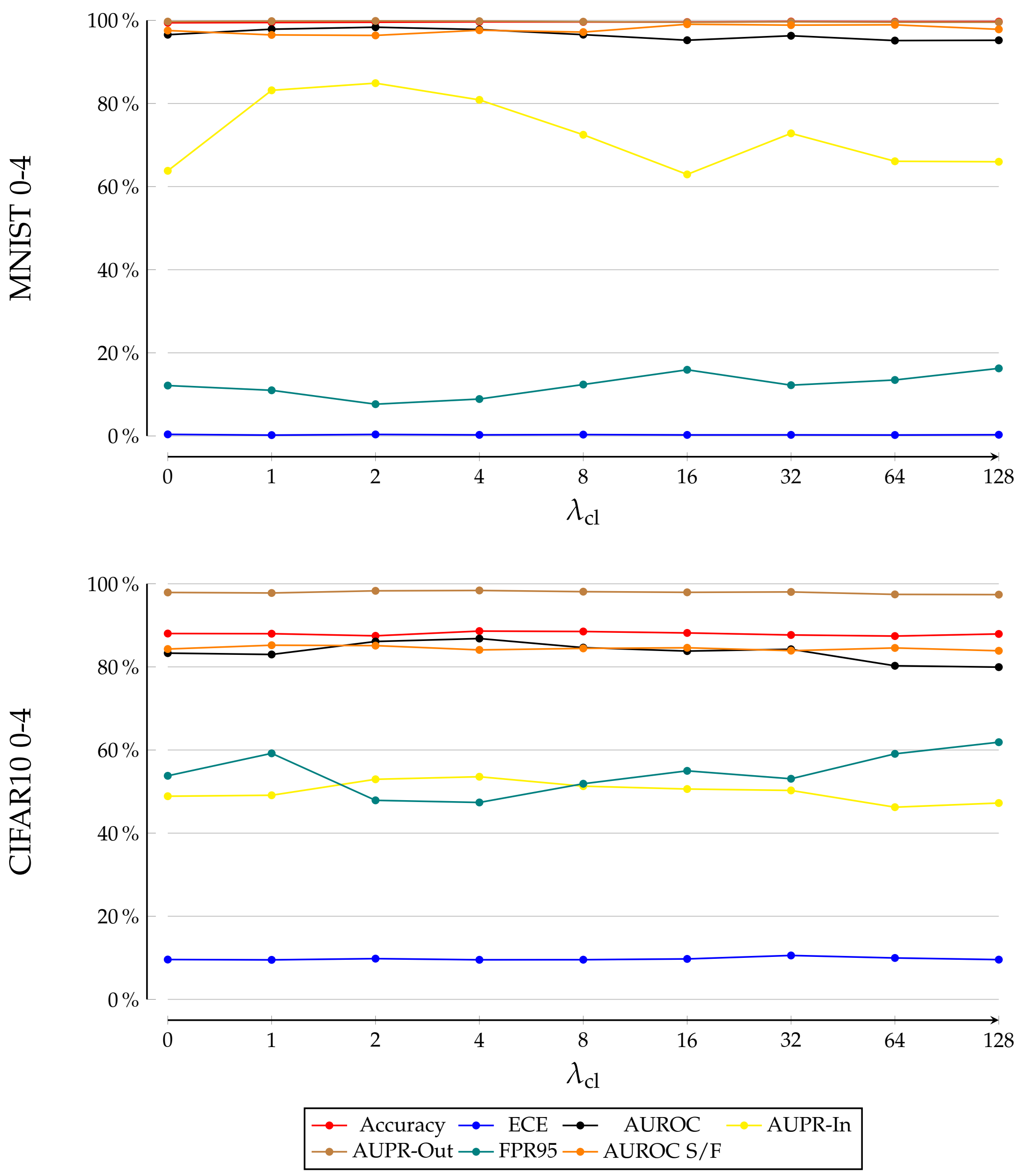

Figure B.3: Hyperparameter evaluation for λ_{cl} from Eq. (6.28). All training runs were performed on the respective training dataset and evaluated on the evaluation sets. A fixed seed was used in order to ensure equal conditions for all parameter combinations. For MNIST 0-4 the other hyperparameters were fixed at $\lambda_R = 14$, $\lambda_{\text{real}} = 0.5$, latent dimension $= 16$, while for CIFAR10 0-4 they were fixed at $\lambda_R = 0$, $\lambda_{\text{real}} = 0.6$ and latent dimension $= 128$. The OoD detection results were computed on the evaluation sets of the respective OoD datasets from Table 8.2.

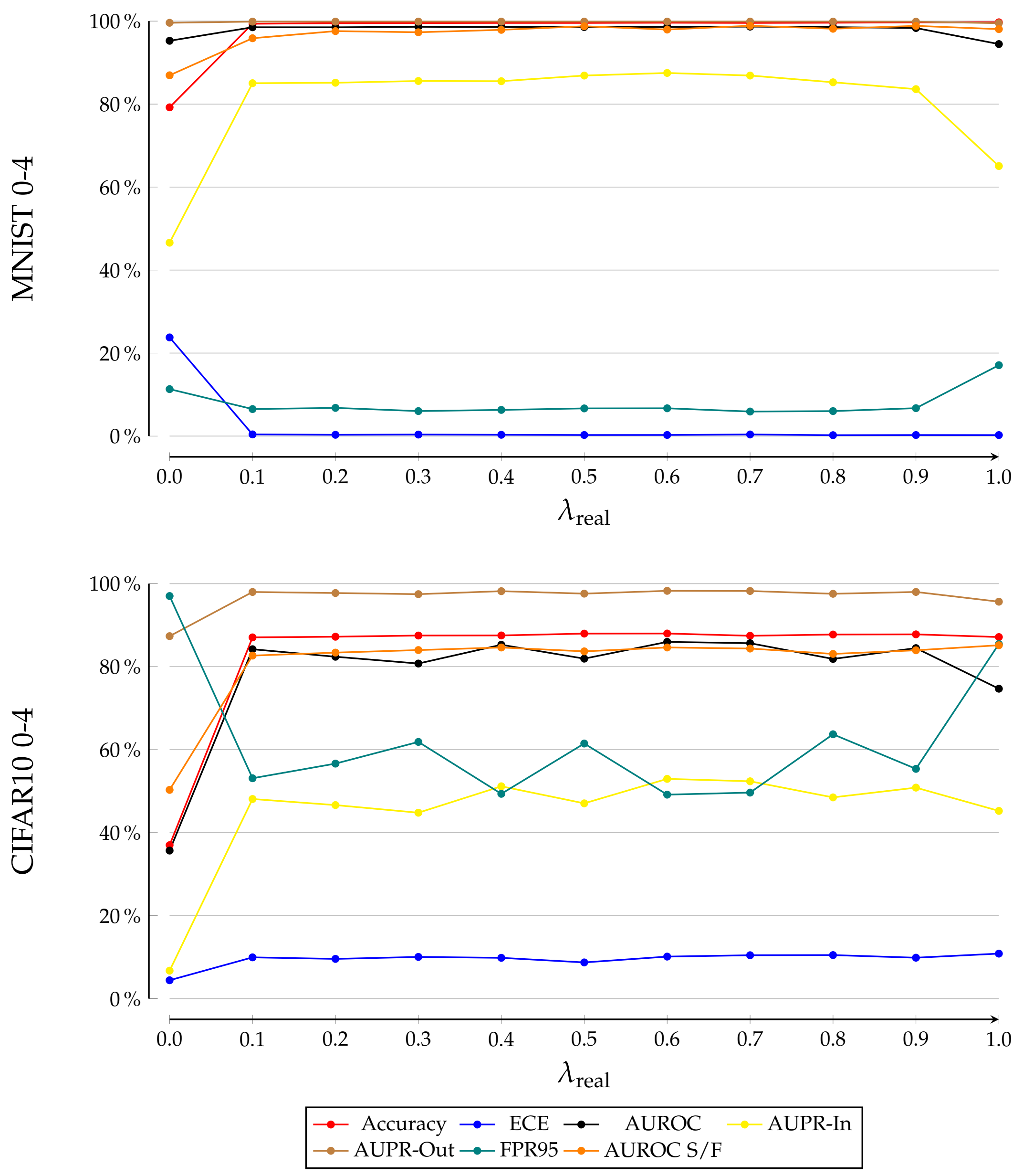

Figure B.4: Hyperparameter evaluation for λ_{real} from Eq. (6.30). All training runs were performed on the respective training dataset and evaluated on the evaluation sets. A fixed seed was used in order to ensure equal conditions for all parameter combinations. For MNIST 0-4 the other hyperparameters were fixed at $\lambda_R = 32$, $\lambda_{\text{cl}} = 1$, latent dimension $= 16$, while for CIFAR10 0-4 they were fixed at $\lambda_R = 0$, $\lambda_{\text{cl}} = 2$ and latent dimension $= 128$. The OoD detection results were computed on the evaluation sets of the respective OoD datasets from Table 8.2.

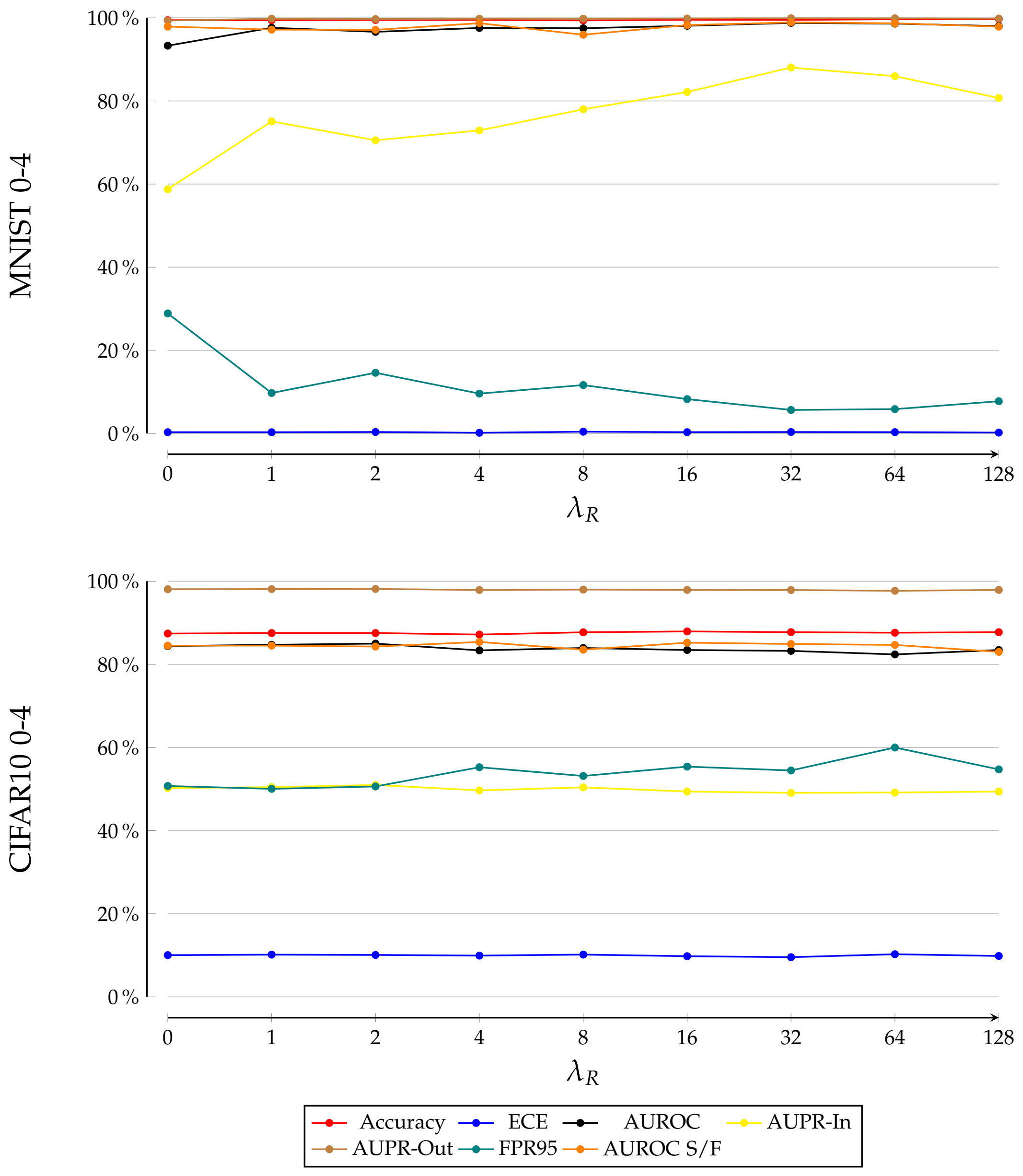

Figure B.5: Hyperparameter evaluation for λ_R from Eq. (6.28). All training runs were performed on the respective training dataset and evaluated on the evaluation sets. A fixed seed was used in order to ensure equal conditions for all parameter combinations. For MNIST 0-4 the other hyperparameters were fixed at $\lambda_R = 32$, $\lambda_{\text{cl}} = 1$, $\lambda_{\text{real}} = 0.5$, while for CIFAR10 0-4 they were fixed at $\lambda_R = 0$, $\lambda_{\text{cl}} = 4$ and $\lambda_{\text{real}} = 0.6$. The OoD detection results were computed on the evaluation sets of the respective OoD datasets from Table 8.2.

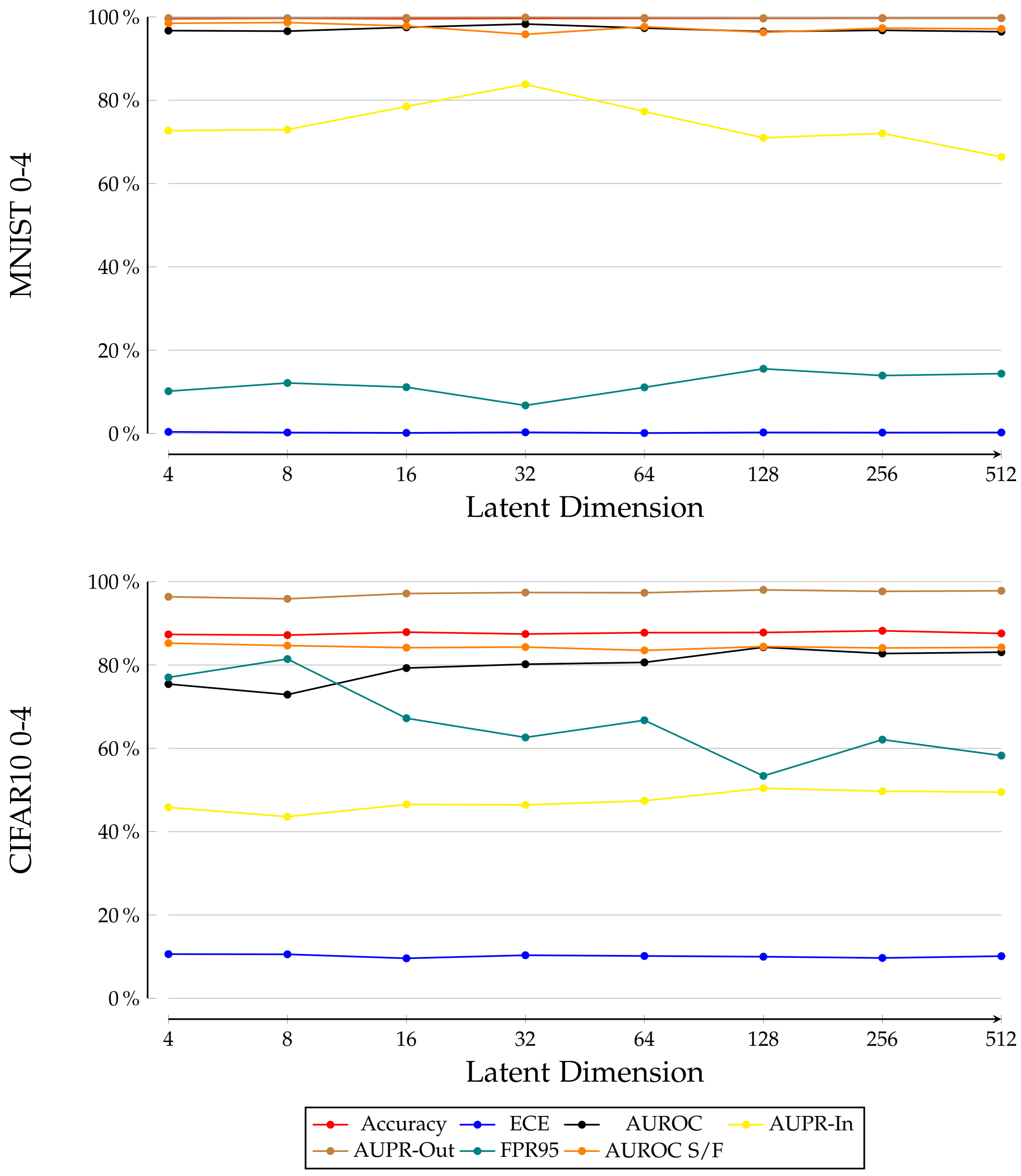

Figure B.6: Hyperparameter evaluation for the latent dimension of the cAE in Eq. (6.27). All training runs were performed on the respective training dataset and evaluated on the evaluation sets. A fixed seed was used in order to ensure equal conditions for all parameter combinations. For MNIST 0-4 the other hyperparameters were fixed at $\lambda_{\mathrm{cl}} = 1$, $\lambda_{\mathrm{real}} = 0.5$ and latent dimension=16, while for CIFAR10 0-4 they were fixed at $\lambda_{\mathrm{cl}} = 4$ and $\lambda_{\mathrm{real}} = 0.6$ and latent dimension=128. The OoD detection results were computed on the evaluation sets of the respective OoD datasets from Table 8.2.

BIBLIOGRAPHY

[1] Moloud Abdar, Farhad Pourpanah, Sadiq Hussain, Dana Rezazadegan, Li Liu, Mohammad Ghavamzadeh, Paul W. Fieguth, Xiaochun Cao, Abbas Khosravi, U. Rajendra Acharya, Vladimir Makarenkov, and Saeid Nahavandi. "A review of uncertainty quantification in deep learning: Techniques, applications and challenges." In: *Information Fusion* 76 (2021), pp. 243–297. DOI: 10.1016/j.inffus.2021.05.008.

[2] Charu C. Aggarwal, Alexander Hinneburg, and Daniel A. Keim. "On the Surprising Behavior of Distance Metrics in High Dimensional Spaces." In: *Database Theory - ICDT 2001, 8th International Conference, London, UK, January 4-6, 2001, Proceedings*. Ed. by Jan Van den Bussche and Victor Vianu. Vol. 1973. Lecture Notes in Computer Science. Springer, 2001, pp. 420–434. DOI: 10.1007/3-540-44503-X_27.

[3] Ahmed Aldahdooh, Wassim Hamidouche, Sid Ahmed Fezza, and Olivier Déforges. "Adversarial example detection for DNN models: a review and experimental comparison." In: *Artificial Intelligence Review* 55.6 (2022), pp. 4403–4462. DOI: 10.1007/s10462-021-10125-w.

[4] Martin Arjovsky, Soumith Chintala, and Léon Bottou. "Wasserstein generative adversarial networks." In: *International conference on machine learning*. PMLR. 2017, pp. 214–223.

[5] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation." In: *PloS one* 10.7 (2015), e0130140.

[6] David Baehrens, Timon Schroeter, Stefan Harmeling, Motoaki Kawanabe, Katja Hansen, and Klaus-Robert Müller. "How to Explain Individual Classification Decisions." In: *The Journal of Machine Learning Research* 11 (2010), pp. 1803–1831. DOI: 10.5555/1756006.1859912.

[7] Kevin S. Beyer, Jonathan Goldstein, Raghu Ramakrishnan, and Uri Shaft. "When Is "Nearest Neighbor" Meaningful?" In: *Database Theory - ICDT '99, 7th International Conference, Jerusalem, Israel, January 10-12, 1999, Proceedings*. Ed. by Catriel Beeri and Peter Buneman. Vol. 1540. Lecture Notes in Computer Science. Springer, 1999, pp. 217–235. DOI: 10.1007/3-540-49257-7_15.

[8] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer, 2006, p. 758. ISBN: 9781493938438. DOI: 10.1007/978-0-387-45528-0.

[9] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. "Weight uncertainty in neural network." In: *International conference on machine learning*. PMLR. 2015, pp. 1613–1622.

[10] Léon Bottou. "Stochastic Gradient Learning in Neural Networks." In: *Proceedings of Neuro-Nîmes 91*. Nimes, France: EC2, 1991.

[11] Leo Breiman. "Bagging Predictors." In: *Machine Learning* 24.2 (1996), pp. 123–140. DOI: 10.1007/BF00058655.

[12] Andrew Brock, Jeff Donahue, and Karen Simonyan. "Large Scale GAN Training for High Fidelity Natural Image Synthesis." In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[13] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D. Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. "Language models are few-shot learners." In: *Advances in neural information processing systems* 33 (2020), pp. 1877–1901.

[14] Thomas Bühler and Matthias Hein. "Spectral clustering based on the graph p-Laplacian." In: *Proceedings of the 26th Annual International Conference on Machine Learning, ICML 2009, Montreal, Quebec, Canada, June 14-18, 2009*. Ed. by Andrea Pohoreckyj Danyluk, Léon Bottou, and Michael L. Littman. Vol. 382. ACM International Conference Proceeding Series. ACM, 2009, pp. 81–88. DOI: 10.1145/1553374.1553385.

[15] Nicholas Carlini and David A. Wagner. "Towards Evaluating the Robustness of Neural Networks." In: *2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017*. IEEE Computer Society, 2017, pp. 39–57. DOI: 10.1109/SP.2017.49.

[16] Robin Chan, Krzysztof Lis, Svenja Uhlemeyer, Hermann Blum, Sina Honari, Roland Siegwart, Pascal Fua, Mathieu Salzmann, and Matthias Rottmann. "SegmentMeIfYouCan: A Benchmark for Anomaly Segmentation." In: *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*. Ed. by Joaquin Vanschoren and Sai-Kit Yeung. 2021.

[17] Haw-Shiuan Chang, Erik G. Learned-Miller, and Andrew McCallum. "Active Bias: Training More Accurate Neural Networks by Emphasizing High Variance Samples." In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. Ed. by Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett. 2017, pp. 1002–1012.

[18] Bertrand Charpentier, Daniel Zügner, and Stephan Günnemann. "Posterior Network: Uncertainty Estimation without OOD Samples via Density-Based Pseudo-Counts." In: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020,*

December 6-12, 2020, virtual. Ed. by Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin. 2020.

[19] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andre Van Schaik. "EMNIST: Extending MNIST to handwritten letters." In: *2017 international joint conference on neural networks (IJCNN)*. IEEE. 2017, pp. 2921–2926.

[20] Charles Corbière, Nicolas Thome, Antoine Saporta, Tuan-Hung Vu, Matthieu Cord, and Patrick Pérez. "Confidence Estimation via Auxiliary Models." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44.10 (2022), pp. 6043–6055. DOI: 10.1109/TPAMI.2021.3085983.

[21] Corinna Cortes, Giulia DeSalvo, and Mehryar Mohri. "Learning with Rejection." In: *Algorithmic Learning Theory - 27th International Conference, ALT 2016, Bari, Italy, October 19-21, 2016, Proceedings*. Ed. by Ronald Ortner, Hans Ulrich Simon, and Sandra Zilles. Vol. 9925. Lecture Notes in Computer Science. 2016, pp. 67–82. DOI: 10.1007/978-3-319-46379-7_5.

[22] Haskell B. Curry. "The Method of Steepest Descent For Non-Linear Minimization Problems." In: *Quarterly of Applied Mathematics* 2.3 (1944), pp. 258–261. ISSN: 0033569X, 15524485.

[23] Navneet Dalal and Bill Triggs. "Histograms of oriented gradients for human detection." In: *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*. Vol. 1. Ieee. 2005, pp. 886–893.

[24] Sina Däubener, Lea Schönherr, Asja Fischer, and Dorothea Kolossa. "Detecting Adversarial Examples for Speech Recognition via Uncertainty Quantification." In: *Interspeech 2020, 21st Annual Conference of the International Speech Communication Association, Virtual Event, Shanghai, China, 25-29 October 2020*. Ed. by Helen Meng, Bo Xu, and Thomas Fang Zheng. ISCA, 2020, pp. 4661–4665. DOI: 10.21437/Interspeech.2020-2734.

[25] Jesse Davis and Mark Goadrich. "The relationship between Precision-Recall and ROC curves." In: *Machine Learning, Proceedings of the Twenty-Third International Conference (ICML 2006), Pittsburgh, Pennsylvania, USA, June 25-29, 2006*. Ed. by William W. Cohen and Andrew W. Moore. Vol. 148. ACM International Conference Proceeding Series. ACM, 2006, pp. 233–240. DOI: 10.1145/1143844.1143874.

[26] Morris H. DeGroot and Stephen E. Fienberg. "The Comparison and Evaluation of Forecasters." In: *Journal of the Royal Statistical Society. Series D (The Statistician)* 32.1/2 (1983), pp. 12–22. ISSN: 00390526, 14679884.

[27] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. "ImageNet: A large-scale hierarchical image database." In: *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*. IEEE Computer Society, 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848.

[28] Stefan Depeweg, José Miguel Hernández-Lobato, Finale Doshi-Velez, and Steffen Udluft. "Decomposition of Uncertainty in Bayesian Deep Learning for Efficient and Risk-sensitive Learning." In: *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*. Ed. by Jennifer G. Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, 2018, pp. 1192–1201.

[29] Terrance DeVries and Graham W. Taylor. "Learning Confidence for Out-of-Distribution Detection in Neural Networks." In: *CoRR* abs/1802.04865 (2018).

[30] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. "Boosting Adversarial Attacks With Momentum." In: *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*. Computer Vision Foundation / IEEE Computer Society, 2018, pp. 9185–9193. DOI: 10.1109/CVPR.2018.00957.

[31] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale." In: *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[32] Harris Drucker, Robert E. Schapire, and Patrice Y. Simard. "Improving Performance in Neural Networks Using a Boosting Algorithm." In: *Advances in Neural Information Processing Systems 5, [NIPS Conference, Denver, Colorado, USA, November 30 - December 3, 1992]*. Ed. by Stephen Jose Hanson, Jack D. Cowan, and C. Lee Giles. Morgan Kaufmann, 1992, pp. 42–49.

[33] John C. Duchi, Elad Hazan, and Yoram Singer. "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization." In: *Journal of Machine Learning Research* 12 (2011), pp. 2121–2159. DOI: 10.5555/1953048.2021068.

[34] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. "Robust Physical-World Attacks on Deep Learning Visual Classification." In: *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*. Computer Vision Foundation / IEEE Computer Society, 2018, pp. 1625–1634. DOI: 10.1109/CVPR.2018.00175.

[35] Gianni Franchi, Andrei Bursuc, Emanuel Aldea, Séverine Dubuisson, and Isabelle Bloch. "One Versus All for Deep Neural Network for Uncertainty (OVNNI) Quantification." In: *Conference on Neural Information Processing Systems. Workshop on Bayesian Deep Learning*. 2020.

[36] Yarin Gal and Zoubin Ghahramani. "Bayesian Convolutional Neural Networks with Bernoulli Approximate Variational Inference." In: *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Workshop Track Proceedings*. 2016.

[37] Yarin Gal and Zoubin Ghahramani. "Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning." In: *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*. Ed. by Maria-Florina Balcan and Kilian Q. Weinberger. Vol. 48. JMLR Workshop and Conference Proceedings. JMLR.org, 2016, pp. 1050–1059.

[38] Jakob Gawlikowski, Cedrique Rovile Njieutcheu Tassi, Mohsin Ali, Jongseok Lee, Matthias Humt, Jianxiang Feng, Anna M. Kruspe, Rudolph Triebel, Peter Jung, Ribana Roscher, Muhammad Shahzad, Wen Yang, Richard Bamler, and Xiao Xiang Zhu. "A Survey of Uncertainty in Deep Neural Networks." In: *CoRR* abs/2107.03342 (2021).

[39] Yonatan Geifman and Ran El-Yaniv. "Selective Classification for Deep Neural Networks." In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. Ed. by Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett. 2017, pp. 4878–4887.

[40] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. "Deep Sparse Rectifier Neural Networks." In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011, Fort Lauderdale, USA, April 11-13, 2011*. Ed. by Geoffrey J. Gordon, David B. Dunson, and Miroslav Dudik. Vol. 15. JMLR Proceedings. JMLR.org, 2011, pp. 315–323.

[41] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[42] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. "Generative Adversarial Nets." In: *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*. Ed. by Zoubin Ghahramani, Max Welling, Corinna Cortes, Neil D. Lawrence, and Kilian Q. Weinberger. 2014, pp. 2672–2680.

[43] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. "Explaining and Harnessing Adversarial Examples." In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2015.

[44] Kathrin Grosse, David Pfaff, Michael T. Smith, and Michael Backes. "The Limitations of Model Uncertainty in Adversarial Settings." In: *Conference on Neural Information Processing Systems. Workshop on Bayesian Deep Learning*. 2019.

[45] Dayan Guan, Jiaxing Huang, Aoran Xiao, Shijian Lu, and Yanpeng Cao. "Uncertainty-Aware Unsupervised Domain Adaptation in Object Detection." In: *IEEE Transactions on Multimedia* 24 (2022), pp. 2502–2514. DOI: `10.1109/TMM.2021.3082687`.

[46] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C. Courville. "Improved Training of Wasserstein GANs." In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. Ed. by Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett. 2017, pp. 5767–5777.

[47] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. "On Calibration of Modern Neural Networks." In: *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, 2017, pp. 1321–1330.

[48] Songqiao Han, Xiyang Hu, Hailiang Huang, Minqi Jiang, and Yue Zhao. "ADBench: Anomaly Detection Benchmark." In: *Advances in Neural Information Processing Systems*. Ed. by S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh. Vol. 35. Curran Associates, Inc., 2022, pp. 32142–32159.

[49] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37.9 (2015), pp. 1904–1916. DOI: 10.1109/TPAMI.2015.2389824.

[50] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep Residual Learning for Image Recognition." In: *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. IEEE Computer Society, 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90.

[51] Matthias Hein, Maksym Andriushchenko, and Julian Bitterwolf. "Why ReLU Networks Yield High-Confidence Predictions Far Away From the Training Data and How to Mitigate the Problem." In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*. Computer Vision Foundation / IEEE, 2019, pp. 41–50. DOI: 10.1109/CVPR.2019.00013.

[52] Dan Hendrycks and Kevin Gimpel. "A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks." In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[53] Dan Hendrycks, Mantas Mazeika, and Thomas G. Dietterich. "Deep Anomaly Detection with Outlier Exposure." In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[54] Irina Higgins, Loic Matthey, Arka Pal, Christopher P. Burgess, Xavier Glorot, Matthew M. Botvinick, Shakir Mohamed, and Alexander Lerchner. "beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework." In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[55] Julia Hornauer and Vasileios Belagiannis. "Gradient-Based Uncertainty for Monocular Depth Estimation." In: *Computer Vision - ECCV 2022 - 17th European Conference, Tel Aviv, Israel, October 23-27, 2022, Proceedings, Part XX*. Ed. by Shai Avidan, Gabriel J. Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner. Vol. 13680. Lecture Notes in Computer Science. Springer, 2022, pp. 613–630. DOI: 10.1007/978-3-031-20044-1_35.

[56] Kurt Hornik, Maxwell B. Stinchcombe, and Halbert White. "Multilayer feedforward networks are universal approximators." In: *Neural Networks* 2.5 (1989), pp. 359–366. DOI: 10.1016/0893-6080(89)90020-8.

[57] Yen-Chang Hsu, Yilin Shen, Hongxia Jin, and Zsolt Kira. "Generalized ODIN: Detecting Out-of-Distribution Image Without Learning From Out-of-Distribution Data." In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*. Computer Vision Foundation / IEEE, 2020, pp. 10948–10957. DOI: 10.1109/CVPR42600.2020.01096.

[58] Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E. Hopcroft, and Kilian Q. Weinberger. "Snapshot Ensembles: Train 1, Get M for Free." In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[59] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. "Densely Connected Convolutional Networks." In: *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. IEEE Computer Society, 2017, pp. 2261–2269. DOI: 10.1109/CVPR.2017.243.

[60] Rui Huang, Andrew Geng, and Yixuan Li. "On the Importance of Gradients for Detecting Distributional Shifts in the Wild." In: *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*. Ed. by Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan. 2021, pp. 677–689.

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

[62] Badr Youbi Idrissi, Diane Bouchacourt, Randall Balestriero, Ivan Evtimov, Caner Hazirbas, Nicolas Ballas, Pascal Vincent, Michal Drozdzal, David Lopez-Paz, and Mark Ibrahim. "ImageNet-X: Understanding Model Mistakes with Factor of Variation Annotations." In: *CoRR* abs/2211.01866 (2022). DOI: 10.48550/arXiv.2211.01866.

[63] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry P. Vetrov, and Andrew Gordon Wilson. "Averaging Weights Leads to Wider Optima and Better Generalization." In: *Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence, UAI 2018, Monterey, California, USA,*

August 6-10, 2018. Ed. by Amir Globerson and Ricardo Silva. AUAI Press, 2018, pp. 876–885.

[64] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization." In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2015.

[65] Diederik P. Kingma and Max Welling. "Auto-Encoding Variational Bayes." In: *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2014.

[66] Armen Der Kiureghian and Ove Ditlevsen. "Aleatory or epistemic? Does it matter?" In: *Structural Safety* 31.2 (Mar. 2009), pp. 105–112. DOI: `10.1016/j.strusafe.2008.06.020`.

[67] Shu Kong and Deva Ramanan. "OpenGAN: Open-Set Recognition via Open Data Generation." In: *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*. IEEE, 2021, pp. 793–802. DOI: `10.1109/ICCV48922.2021.00085`.

[68] Alex Krizhevsky. *Learning Multiple Layers of Features from Tiny Images*. Tech. rep. University of Toronto, 2009.

[69] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks." In: *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*. Ed. by Peter L. Bartlett, Fernando C. N. Pereira, Christopher J. C. Burges, Léon Bottou, and Kilian Q. Weinberger. 2012, pp. 1106–1114.

[70] Anders Krogh and John A. Hertz. "A Simple Weight Decay Can Improve Generalization." In: *Advances in Neural Information Processing Systems 4, [NIPS Conference, Denver, Colorado, USA, December 2-5, 1991]*. Ed. by John E. Moody, Stephen Jose Hanson, and Richard Lippmann. Morgan Kaufmann, 1991, pp. 950–957.

[71] Solomon Kullback and Richard A. Leibler. "On Information and Sufficiency." In: *The Annals of Mathematical Statistics* 22.1 (Mar. 1951), pp. 79–86. DOI: `10.1214/aoms/1177729694`.

[72] Ananya Kumar, Percy Liang, and Tengyu Ma. "Verified Uncertainty Calibration." In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*. Ed. by Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett. 2019, pp. 3787–3798.

[73] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. "Adversarial examples in the physical world." In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings*. OpenReview.net, 2017.

[74] Brenden M. Lake, Ruslan Salakhutdinov, and Joshua B. Tenenbaum. "Human-level concept learning through probabilistic program induction." In: *Science* 350 (2015), pp. 1332–1338. ISSN: 0036-8075. DOI: `10.1126/science.aab3050`.

[75] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. "Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles." In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. Ed. by Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett. 2017, pp. 6402–6413.

[76] Ya Le and Xuan Yang. "Tiny imagenet visual recognition challenge." In: *CS 231N* 7.7 (2015), p. 3.

[77] Yann Lecun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. "Gradient-based learning applied to document recognition." In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324. DOI: `10.1109/5.726791`.

[78] Jinsol Lee and Ghassan AlRegib. "Gradients as a Measure of Uncertainty in Neural Networks." In: *IEEE International Conference on Image Processing, ICIP 2020, Abu Dhabi, United Arab Emirates, October 25-28, 2020*. IEEE, 2020, pp. 2416–2420. DOI: `10.1109/ICIP40778.2020.9190679`.

[79] Kimin Lee, Honglak Lee, Kibok Lee, and Jinwoo Shin. "Training Confidence-calibrated Classifiers for Detecting Out-of-Distribution Samples." In: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[80] Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. "A Simple Unified Framework for Detecting Out-of-Distribution Samples and Adversarial Attacks." In: *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*. Ed. by Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett. 2018, pp. 7167–7177.

[81] David D. Lewis and William A. Gale. "A Sequential Algorithm for Training Text Classifiers." In: *Proceedings of the 17th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval. Dublin, Ireland, 3-6 July 1994 (Special Issue of the SIGIR Forum)*. Ed. by W. Bruce Croft and C. J. van Rijsbergen. ACM/Springer, 1994, pp. 3–12. DOI: `10.1007/978-1-4471-2099-5_1`.

[82] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. "Visualizing the Loss Landscape of Neural Nets." In: *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*. Ed. by Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett. 2018, pp. 6391–6401.

[83] Shiyu Liang, Yixuan Li, and R. Srikant. "Enhancing The Reliability of Out-of-distribution Image Detection in Neural Networks." In: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[84] Dong C. Liu and Jorge Nocedal. "On the limited memory BFGS method for large scale optimization." In: *Mathematical Programming* 45.1-3 (1989), pp. 503–528. DOI: 10.1007/BF01589116.

[85] David G. Lowe. "Distinctive image features from scale-invariant keypoints." In: *International journal of computer vision* 60.2 (2004), pp. 91–110.

[86] Kira Maag and Tobias Riedlinger. "Pixel-wise Gradient Uncertainty for Convolutional Neural Networks applied to Out-of-Distribution Segmentation." In: *CoRR* abs/2303.06920 (Mar. 2023). DOI: 10.48550/arXiv.2303.06920.

[87] Wesley J. Maddox, Pavel Izmailov, Timur Garipov, Dmitry P. Vetrov, and Andrew Gordon Wilson. "A Simple Baseline for Bayesian Uncertainty in Deep Learning." In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*. Ed. by Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett. 2019, pp. 13132–13143.

[88] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. "Towards Deep Learning Models Resistant to Adversarial Attacks." In: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[89] Andrey Malinin and Mark J. F. Gales. "Predictive Uncertainty Estimation via Prior Networks." In: *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*. Ed. by Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett. 2018, pp. 7047–7058.

[90] Warren S. McCulloch and Walter Pitts. "A logical calculus of the ideas immanent in nervous activity." In: *The Bulletin of Mathematical Biophysics* 5.4 (Dec. 1943), pp. 115–133. DOI: 10.1007/bf02478259.

[91] George A. Miller. *WordNet: An electronic lexical database*. MIT press, 1998.

[92] M. Minsky and S. Papert. *Perceptrons: An Introduction to Computational Geometry*. MIT Press, 1969. ISBN: 9780262630221.

[93] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. "DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks." In: *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. IEEE Computer Society, 2016, pp. 2574–2582. DOI: 10.1109/CVPR.2016.282.

[94] Mahdi Pakdaman Naeini, Gregory F. Cooper, and Milos Hauskrecht. "Obtaining Well Calibrated Probabilities Using Bayesian Binning." In: *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*. Ed. by Blai Bonet and Sven Koenig. AAAI Press, 2015, pp. 2901–2907.

[95] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. "Reading Digits in Natural Images with Unsupervised Feature Learning." In: *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*. 2011.

[96] Cuong Phuc Ngo, Amadeus Aristo Winarto, Connie Khor Li Kou, Sojeong Park, Farhan Akram, and Hwee Kuan Lee. "Fence GAN: Towards Better Anomaly Detection." In: *31st IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2019, Portland, OR, USA, November 4-6, 2019*. IEEE, 2019, pp. 141–148. DOI: 10.1109/ICTAI.2019.00028.

[97] Vu-Linh Nguyen, Mohammad Hossein Shaker, and Eyke Hüllermeier. "How to measure uncertainty in uncertainty sampling for active learning." In: *Machine Learning* 111.1 (2022), pp. 89–122. DOI: 10.1007/s10994-021-06003-9.

[98] Philipp Oberdiek, Gernot Fink, and Matthias Rottmann. "UQGAN: A Unified Model for Uncertainty Quantification of Deep Classifiers trained via Conditional GANs." In: *Advances in Neural Information Processing Systems*. Ed. by S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh. Vol. 35. Curran Associates, Inc., 2022, pp. 21371–21385.

[99] Philipp Oberdiek, Matthias Rottmann, and Gernot A. Fink. "Detection and Retrieval of Out-of-Distribution Objects in Semantic Segmentation." In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. IEEE, June 2020. DOI: 10.1109/cvprw50498.2020.00172.

[100] Philipp Oberdiek, Matthias Rottmann, and Hanno Gottschalk. "Classification Uncertainty of Deep Neural Networks Based on Gradient Information." In: *Artificial Neural Networks in Pattern Recognition*. Springer International Publishing, 2018, pp. 113–125. DOI: 10.1007/978-3-319-99978-4_9.

[101] Timo Ojala, Matti Pietikäinen, and David Harwood. "A comparative study of texture measures with classification based on featured distributions." In: *Pattern recognition* 29.1 (1996), pp. 51–59.

[102] Shreyas Padhy, Zachary Nado, Jie Ren, Jeremiah Liu, Jasper Snoek, and Balaji Lakshminarayanan. "Revisiting One-vs-All Classifiers for Predictive Uncertainty and Out-of-Distribution Detection in Neural Networks." In: *37th International Conference on Machine Learning, ICML 2020, Workshop on*

Uncertainty and Robustness in Deep Learning, Vienna, Austria, July 12-18. July 2020.

[103] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Z. Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. "PyTorch: An Imperative Style, High-Performance Deep Learning Library." In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*. Ed. by Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett. 2019, pp. 8024–8035.

[104] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. "Hierarchical Text-Conditional Image Generation with CLIP Latents." In: *CoRR* abs/2204.06125 (2022). DOI: 10.48550/arXiv.2204.06125.

[105] Jie Ren, Peter J. Liu, Emily Fertig, Jasper Snoek, Ryan Poplin, Mark A. DePristo, Joshua V. Dillon, and Balaji Lakshminarayanan. "Likelihood Ratios for Out-of-Distribution Detection." In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*. Ed. by Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett. 2019, pp. 14680–14691.

[106] Tobias Riedlinger, Matthias Rottmann, Marius Schubert, and Hanno Gottschalk. "Gradient-based quantification of epistemic uncertainty for deep object detectors." In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2023, pp. 3921–3931.

[107] Frank Rosenblatt. "The perceptron: A probabilistic model for information storage and organization in the brain." In: *Psychological Review* 65.6 (1958), pp. 386–408. DOI: 10.1037/h0042519.

[108] Matthias Rottmann, Pascal Colling, Thomas-Paul Hack, Robin Chan, Fabian Hüger, Peter Schlicht, and Hanno Gottschalk. "Prediction Error Meta Classification in Semantic Segmentation: Detection via Aggregated Dispersion Measures of Softmax Probabilities." In: *2020 International Joint Conference on Neural Networks, IJCNN 2020, Glasgow, United Kingdom, July 19-24, 2020*. IEEE, 2020, pp. 1–9. DOI: 10.1109/IJCNN48605.2020.9206659.

[109] Andras Rozsa, Ethan M. Rudd, and Terrance E. Boult. "Adversarial Diversity and Hard Positive Generation." In: *2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2016, Las Vegas, NV, USA, June 26 - July 1, 2016*. IEEE Computer Society, 2016, pp. 410–417. DOI: 10.1109/CVPRW.2016.58.

[110] David E. Rumelhart and James L. McClelland. "Learning Internal Representations by Error Propagation." In: *Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Foundations*. 1987, pp. 318–362.

[111] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L. Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, Jonathan Ho, David J. Fleet, and Mohammad Norouzi. "Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding." In: *Advances in Neural Information Processing Systems*. Ed. by S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh. Vol. 35. Curran Associates, Inc., 2022, pp. 36479–36494.

[112] Kuniaki Saito and Kate Saenko. "OVANet: One-vs-All Network for Universal Domain Adaptation." In: *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*. IEEE, 2021, pp. 8980–8989. DOI: 10.1109/ICCV48922.2021.00887.

[113] Takaya Saito and Marc Rehmsmeier. "The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets." In: *PloS one* 10.3 (2015), e0118432.

[114] Christos Sakaridis, Dengxin Dai, and Luc Van Gool. "Map-Guided Curriculum Domain Adaptation and Uncertainty-Aware Evaluation for Semantic Nighttime Image Segmentation." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44.6 (2022), pp. 3139–3153. DOI: 10.1109/TPAMI.2020.3045882.

[115] Robert E. Schapire. "The Strength of Weak Learnability." In: *Machine Learning* 5 (1990), pp. 197–227. DOI: 10.1007/BF00116037.

[116] Thomas Schlegl, Philipp Seeböck, Sebastian M. Waldstein, Ursula Schmidt-Erfurth, and Georg Langs. "Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery." In: *Information Processing in Medical Imaging - 25th International Conference, IPMI 2017, Boone, NC, USA, June 25-30, 2017, Proceedings*. Ed. by Marc Niethammer, Martin Styner, Stephen R. Aylward, Hongtu Zhu, Ipek Oguz, Pew-Thian Yap, and Dinggang Shen. Vol. 10265. Lecture Notes in Computer Science. Springer, 2017, pp. 146–157. DOI: 10.1007/978-3-319-59050-9_12.

[117] Murat Sensoy, Lance M. Kaplan, Federico Cerutti, and Maryam Saleki. "Uncertainty-Aware Deep Classifiers Using Generative Models." In: *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*. AAAI Press, 2020, pp. 5620–5627.

[118] Murat Sensoy, Lance M. Kaplan, and Melih Kandemir. "Evidential Deep Learning to Quantify Classification Uncertainty." In: *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*. Ed. by Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett. 2018, pp. 3183–3193.

[119] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning - From Theory to Algorithms*. Cambridge University Press, 2014. ISBN: 978-1-10-705713-5.

[120] Claude E. Shannon. "A Mathematical Theory of Communication." In: *Bell System Technical Journal* 27.3 (July 1948), pp. 379–423. DOI: 10.1002/j.1538-7305.1948.tb01338.x.

[121] Mahmood Sharif, Sruti Bhagavatula, Lujo Bauer, and Michael K. Reiter. "Accessorize to a Crime: Real and Stealthy Attacks on State-of-the-Art Face Recognition." In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*. Ed. by Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi. ACM, 2016, pp. 1528–1540. DOI: 10.1145/2976749.2978392.

[122] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition." In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2015.

[123] Lewis Smith and Yarin Gal. "Understanding Measures of Uncertainty for Adversarial Example Detection." In: *Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence, UAI 2018, Monterey, California, USA, August 6-10, 2018*. Ed. by Amir Globerson and Ricardo Silva. AUAI Press, 2018, pp. 560–569.

[124] Kumar Sricharan and Ashok Srivastava. "Building robust classifiers through generation of confident out of distribution examples." In: *Conference on Neural Information Processing Systems. Workshop on Bayesian Deep Learning*. 2018.

[125] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. "Dropout: a simple way to prevent neural networks from overfitting." In: *The Journal of Machine Learning Research* 15.1 (2014), pp. 1929–1958. DOI: 10.5555/2627435.2670313.

[126] Ke Sun, Zhanxing Zhu, and Zhouchen Lin. "Enhancing the Robustness of Deep Neural Networks by Boundary Conditional GAN." In: *CoRR* abs/1902.11029 (2019).

[127] Ilya Sutskever, James Martens, George E. Dahl, and Geoffrey E. Hinton. "On the importance of initialization and momentum in deep learning." In: *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*. Vol. 28. JMLR Workshop and Conference Proceedings. JMLR.org, 2013, pp. 1139–1147.

[128] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. "Intriguing properties of neural networks." In: *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2014.

[129] Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, YaGuang Li, Hongrae Lee, Huaixiu Steven Zheng, Amin Ghafouri, Marcelo Menegali, Yanping Huang, Maxim Krikun, Dmitry Lepikhin, James Qin, Dehao Chen, Yuanzhong Xu, Zhifeng Chen, Adam Roberts, Maarten Bosma, Yanqi Zhou, Chung-Ching Chang, Igor Krivokon, Will Rusch, Marc Pickett, Kathleen S. Meier-Hellstern, Meredith Ringel Morris, Tulsee Doshi, Renelito Delos Santos, Toju Duke, Johnny Soraker, Ben Zevenbergen, Vinodkumar Prabhakaran, Mark Diaz, Ben Hutchinson, Kristen Olson, Alejandra Molina, Erin Hoffman-John, Josh Lee, Lora Aroyo, Ravi Rajakumar, Alena Butryna, Matthew Lamm, Viktoriya Kuzmina, Joe Fenton, Aaron Cohen, Rachel Bernstein, Ray Kurzweil, Blaise Aguera-Arcas, Claire Cui, Marian Croak, Ed H. Chi, and Quoc Le. "LaMDA: Language Models for Dialog Applications." In: *CoRR* abs/2201.08239 (2022).

[130] Robert Tibshirani. "Regression Shrinkage and Selection via the Lasso." In: *Journal of the Royal Statistical Society. Series B (Methodological)* 58.1 (1996), pp. 267–288. ISSN: 0035-9246.

[131] Tijmen Tieleman and Geoffrey Hinton. *Lecture 6.5 - rmsprop: Divide the gradient by a running average of its recent magnitude*. 2012.

[132] Antonio Torralba, Robert Fergus, and William T. Freeman. "80 Million Tiny Images: A Large Data Set for Nonparametric Object and Scene Recognition." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 30.11 (2008), pp. 1958–1970. DOI: `10.1109/TPAMI.2008.128`.

[133] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian J. Goodfellow, Dan Boneh, and Patrick D. McDaniel. "Ensemble Adversarial Training: Attacks and Defenses." In: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[134] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. "Attention is All you Need." In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. Ed. by Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett. 2017, pp. 5998–6008.

[135] Sachin Vernekar, Ashish Gaurav, Vahdat Abdelzad, Taylor Denouden, Rick Salay, and Krzysztof Czarnecki. "Out-of-distribution Detection in Classifiers via Generation." In: *CoRR* abs/1910.04241 (2019).

[136] Sachin Vernekar, Ashish Gaurav, Taylor Denouden, Buu Phan, Vahdat Abdelzad, Rick Salay, and Krzysztof Czarnecki. "Analysis of Confident-Classifiers for Out-of-distribution Detection." In: *CoRR* abs/1904.12220 (2019).

[137] Cédric Villani. *Optimal transport: old and new*. Vol. 338. Grundlehren der mathematischen Wissenschaften. Springer, 2009.

[138] Max Welling and Yee Whye Teh. "Bayesian Learning via Stochastic Gradient Langevin Dynamics." In: *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*. Ed. by Lise Getoor and Tobias Scheffer. Omnipress, 2011, pp. 681–688.

[139] Yan Xia, Xudong Cao, Fang Wen, Gang Hua, and Jian Sun. "Learning Discriminative Reconstructions for Unsupervised Outlier Removal." In: *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*. IEEE Computer Society, 2015, pp. 1511–1519. DOI: 10.1109/ICCV.2015.177.

[140] Yingda Xia, Dong Yang, Zhiding Yu, Fengze Liu, Jinzheng Cai, Lequan Yu, Zhuotun Zhu, Daguang Xu, Alan L. Yuille, and Holger Roth. "Uncertainty-aware multi-view co-training for semi-supervised medical image segmentation and domain adaptation." In: *Medical Image Analysis* 65 (2020), p. 101766. DOI: 10.1016/j.media.2020.101766.

[141] Han Xiao, Kashif Rasul, and Roland Vollgraf. "Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms." In: *CoRR* abs/1708.07747 (2017).

[142] Jason Yosinski, Jeff Clune, Anh Mai Nguyen, Thomas J. Fuchs, and Hod Lipson. "Understanding Neural Networks Through Deep Visualization." In: *32nd International Conference on Machine Learning, ICML 2015, Deep Learning Workshop, Lille, France, 6-11 July 2015*. 2015.

[143] Fisher Yu, Yinda Zhang, Shuran Song, Ari Seff, and Jianxiong Xiao. "LSUN: Construction of a Large-scale Image Dataset using Deep Learning with Humans in the Loop." In: *CoRR* abs/1506.03365 (2015).

[144] Xiaoyong Yuan, Pan He, Qile Zhu, and Xiaolin Li. "Adversarial Examples: Attacks and Defenses for Deep Learning." In: *IEEE Transactions on Neural Networks and Learning Systems* 30.9 (2019), pp. 2805–2824. DOI: 10.1109/TNNLS.2018.2886017.

[145] Matthew D. Zeiler and Rob Fergus. "Visualizing and Understanding Convolutional Networks." In: *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I*. Ed. by David J. Fleet, Tomás Pajdla, Bernt Schiele, and Tinne Tuytelaars. Vol. 8689. Lecture Notes in Computer Science. Springer, 2014, pp. 818–833. DOI: 10.1007/978-3-319-10590-1_53.

[146] Han Zhang, Ian J. Goodfellow, Dimitris N. Metaxas, and Augustus Odena. "Self-Attention Generative Adversarial Networks." In: *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, 2019, pp. 7354–7363.

[147] David Zimmerer, Peter M. Full, Fabian Isensee, Paul Jäger, Tim Adler, Jens Petersen, Gregor Köhler, Tobias Roß, Annika Reinke, Antanas Kascenas, Bjørn Sand Jensen, Alison Q. O'Neil, Jeremy Tan, Benjamin Hou, James Batten, Huaqi Qiu, Bernhard Kainz, Nina Shvetsova, Irina Fedulova, Dmitry V. Dylov, Baolun Yu, Jianyang Zhai, Jingtao Hu, Runxuan Si, Sihang Zhou, Siqi Wang, Xinyang Li, Xuerun Chen, Yang Zhao, Sergio Naval Marimont, Giacomo Tarroni, Victor Saase, Lena Maier-Hein, and Klaus H. Maier-Hein. "MOOD 2020: A Public Benchmark for Out-of-Distribution Detection and Localization on Medical Images." In: *IEEE Transactions on Medical Imaging* 41.10 (2022), pp. 2728–2738. DOI: `10.1109/TMI.2022.3170077`.

ACRONYMS

AE	Autoencoder
AMT	Amazon Mechanical Turk
AUPR	Area under Precision Recall
AUROC	Area under Receiver Operating Characteristic
BBB	Bayes-by-Backprop
BIM	Basic Iterative Method
BNN	Bayesian Neural Network
cAE	conditional Autoencoder
cGAN	conditional Generative Adversarial Network
CNN	Convolutional Neural Network
DE	Deep Ensemble
DNN	Deep Neural Network
ECE	Expected Calibration Error
ELBO	Evidence Lower Bound
ERM	Empirical Risk Minimizer
FGSM	Fast Gradient Sign Method
FLOP	Floating Point Operation
FP	False Positive
FPR	False Positive Rate
GAN	Generative Adversarial Network
GPU	Graphics Processing Unit
ID	In-Distribution
i.i.d.	independent and identically distributed
ILLC	Iterative Least Likely Class
IoU	Intersection over Union
JS	Jensen-Shannon
KLD	Kullback-Leibler Divergence
LeakyReLU	Leaky Rectified Linear Unit
MAP	Maximum a Posteriori
MC	Monte Carlo
MCD	Monte Carlo-Dropout
MCE	Maximum Calibration Error
MCMC	Markow-Chain-Monte-Carlo
MCP	Maximum Class Probability
MI	Mutual Information
ML	Machine Learning
MLE	Maximum Likelihood Estimation
MLP	Multilayer Perceptron
MRI	Magnetic Resonance Imaging
NLLL	Negative Log-Likelihood Loss
NN	Neural Network
OoC	Out-of-Class
OoD	Out-of-Distribution
OvA	One-versus-All
PGD	Projected Gradient Descent
PR	Precision Recall
QbE	Query by Example
RBF	Radial Basis Function
ReLU	Rectified Linear Unit
ResNet	Residual Neural Network

ROC	Receiver Operating Characteristic	SWAG	SWA-Gaussian
SGD	Stochastic Gradient Descent	TCP	True Class Probability
SPP	Spatial Pyramid Pooling	UQ	Uncertainty Quantification
SVHN	Street View House Numbers	UQGAN	Uncertainty Quantification GAN
SVM	Support Vector Machine	UQGAN-MCD	Uncertainty Quantification GAN - Monte Carlo Dropout
SWA	Stochastic Weight Averaging	VAE	Variational Autoencoder