Department of Mathematics and Computer Science

Miscalibration in Neural Network Classifiers: Causes and Countermeasures

Master Thesis

Nick van de Waterlaat

Supervisors

prof. dr. ir. Wil Michiels (TU/e, NXP)
ir. Geert Derks (NXP)
drs. Frits Schalij (NXP)

Assessment committee

prof. dr. ir. Wil Michiels (TU/e, NXP)
ir. Geert Derks (NXP)
drs. Frits Schalij (NXP)
dr. Sibylle Hess (TU/e)

December 16, 2021

Abstract

In safety-critical applications, it is important for predictions of a neural network classifier to be reliable. A prediction, represented by a probability distribution over classes, is reliable if its probabilities are consistent with the empirical frequencies witnessed from observed outcomes. A classifier that adheres to this property is called well-calibrated. Unfortunately, neural network classifiers often exhibit poor calibration. While numerous ad-hoc techniques have been proposed to reduce the degree of miscalibration, the origination of miscalibration has largely gone unstudied. Based on a decomposition of popular loss functions used to train classifiers, we point toward overfitting and a distribution shift in uncertainty as important causes of miscalibration. Our empirical experiments across different classifiers and datasets support the finding that these phenomena lead to miscalibration. Based on our findings, we introduce techniques for preventing (severe forms of) miscalibration throughout training. Additionally, we outline ad-hoc techniques that reduce the degree of miscalibration after a classifier has been trained. Afterward, we study the effectiveness of these techniques, finding that many techniques can substantially reduce the degree of miscalibration without significantly impacting the classifier's accuracy.

Acknowledgments

First and foremost, I would like to thank my supervisor Wil Michiels for his invaluable guidance, constructive feedback, and all of the engaging discussions we had together. I could not have completed my thesis without his assistance and support. Additionally, I would like to thank my co-supervisors Geert Derks and Frits Schalij for their valuable comments and proofreading while I was working on this thesis. It has been a pleasure working with all of them. Lastly, I would like to express my gratitude to my close friends and family for their support throughout this time.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Advances in deep learning have significantly enhanced the accuracy of neural network classifiers. Consequently, these networks have been employed in complex decision-making processes, such as detecting diseases in medical image analysis [27]. In these cases, neural network classifiers form a crucial component of the entire decision-making process.

Neural network classifiers often predict a probability distribution over its target classes. In many applications, such as safety-critical ones, it is important for these predictions to be reliable. Specifically, the confidence score of each class in the predicted probability distribution should reflect the probability that class has of actually occurring. In practice, this translates to the confidence score of each class aligning with the empirical frequency of that class in observed outcomes. A classifier whose predictions adhere to this property is called *well-calibrated*. Miscalibrated confidence scores can be misleading and unintuitive for decision-making processes as they do not reflect the reality accurately. Therefore, calibration is a desirable, if not important, property in various classification tasks.

The literature proposes numerous techniques that can be applied to predictions of classifiers to improve their calibration. Such mitigative techniques have been studied extensively in the literature. In contrast, the origination of miscalibration has not been studied as well. Nevertheless, discovering possible causes of miscalibration is important as it not only leads to a better understanding of the problem but might also lead to more effective countermeasures against miscalibration.

This work identifies two important causes of miscalibration based on a decomposition of proper scoring rules. Proper scoring rules, such as the log-loss and Brier score, are popular loss functions for training classifiers. Based on a decomposition of these popular loss functions, we point toward overfitting and a distribution shift in uncertainty as important causes of miscalibration. We provide support for these phenomena contributing to a classifier becoming mis-

calibrated through experiments with ResNet classifiers of varying complexities on CIFAR-10 [20], CIFAR-100 [20], and ImageWoof[1]

Based on our findings, we introduce preventative techniques for miscalibration. These techniques reduce the risk that a classifier becomes severely miscalibrated throughout training. Additionally, we list several mitigative techniques for reducing miscalibration after training a classifier. We study the effectiveness of the introduced techniques on ResNet [12] classifiers with varying complexities across CIFAR-10, CIFAR-100, and ImageWoof. We also include results for other popular classifiers, such as LeNet-5 [26], ResNet-110 with Stochastic Depth [16], and DenseNet-40 [15], on CIFAR-10 and CIFAR-100. Our results show that many techniques can substantially reduce the degree of miscalibration. For instance, a straightforward technique called temperature scaling reduced the degree of miscalibration associated with the confidence score of the predicted class by 79% for ResNet-110 on CIFAR-100 without impacting its accuracy.

1.1 Objectives

This thesis addresses three separate but complementary research objectives. First, we would like to detect likely causes of miscalibration in neural network classifiers (research objective 1). Secondly, we want to uncover how miscalibrated confidence scores can be improved concerning calibration (research objective 2). Lastly, we would like to study how different techniques that improve calibration compare to one another (research objective 3).

1.2 Outline

In the following chapters, we introduce the reader to the research domain of calibration by addressing the research objectives formulated in the previous section. A schematic overview of the chapters and their relation to the research objectives is shown in Figure 1.1. We start by addressing the reliability of predictions produced by neural networks classifiers in Chapter 2. In this chapter, we first introduce the notion of calibration informally through various use cases. Subsequently, we formalize the notion of calibration from a mathematical perspective. Based on the mathematical formalization of calibration, we introduce metrics that quantify and estimate the degree of miscalibration in Chapter 3. Afterward, we evaluate the degree of miscalibration across several neural network classifiers with these metrics. Since we find most of them to be miscalibrated, we shift our

[1]The ImageWoof dataset can be found at https://github.com/fastai/imagenette.

focus to the origination of miscalibration (research objective 1) in Chapter 4. Specifically, we hypothesize and confirm two important causes of miscalibration in neural network classifiers. Based on our findings, we propose preventative techniques for miscalibration in Chapter 5. In addition to preventative techniques, we list mitigative techniques for miscalibration in Chapter 6. These chapters focus entirely on techniques for improving miscalibrated confidence scores (research objective 2). We study the effectiveness of these techniques on various classifiers across different datasets in Chapter 7 (research objective 3). We conclude the thesis with a high-level summary, a discussion on the limitations of our work, and an outlook of future work in this area in Chapter 8.

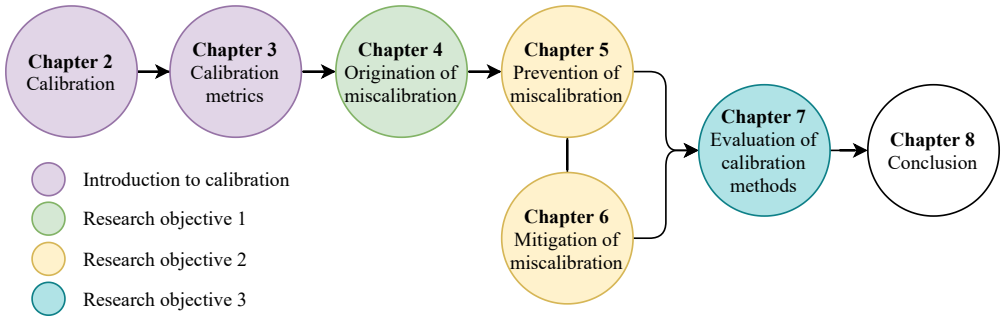


Figure 1.1: Schematic overview of the chapters in this work.

Chapter 2

Calibration

The introduction in the previous chapter briefly provides intuition behind the relevance of calibration. In this chapter, we further introduce the concept of calibration and its significance in machine learning. Specifically, we motivate calibration by presenting several use cases with increasing complexity. Afterward, we formalize the notion of calibration from a mathematical perspective.

2.1 Motivation for calibration

Before diving into the formalization of calibration, we first stress its significance in machine learning. We start with a simple example of a biased coin to build intuition behind measuring the reliability of probability estimates. Afterward, we build upon this example and move towards weather forecasting. Weather forecasting is particularly interesting as it is a traditional but straightforward task where calibration plays a vital role [3]. From weather forecasting, we move towards calibration for classification tasks with a focus on image classification.

2.1.1 Biased coin

Suppose we find a coin on the streets that looks slightly dented. In the case of a *fair* coin, its probability of tossing either head or tail equals 50%. Since our coin looks slightly dented, we strongly doubt it to be fair. Instead, we suspect it to be an *unfair* coin that deviates from the "fair" 50% chance. Suppose that, based on the dent in the coin, we estimate its probability to toss head to equal 75%. We would like to verify whether our estimated probability of tossing head is *reliable*. To that end, we want to determine whether our estimate of tossing head is representative of the actual probability of tossing head. However, the actual probability is generally unknown in practice. Therefore, the best we can do is to estimate it by performing many coin tosses. In other words, we estimate the actual probability based on empirical observations (*i.e.*, realized outcomes of coin tosses). Suppose we perform 1000 coin tosses. Ideally, we would like roughly

75% of these tosses to land head. In that case, our estimated 75% probability of landing head seems reliable. If we find that out of the 1000 tosses, only 270 landed head. Then, our prediction of the coin toss landing head 75% of the time seems improbable and unreliable.

In this example, we could easily verify that the estimated probability of the coin toss landing head was unreliable by performing many coin tosses. However, unlike in this scenario, verifying whether estimated probabilities are reliable based on empirical observations is not always straightforward.

2.1.2 Weather forecasting

Suppose meteorologists at the *Royal Netherlands Meteorological Institute* (KNMI) predict a 70% chance of rain in Eindhoven tomorrow. In other words, the meteorologists are 70% confident that it will rain in Eindhoven tomorrow. How do we know if this confidence estimate of 70% is reliable? In the previous example, we could perform many coin tosses to verify whether our estimated probability aligned with the empirical frequency. In this case, however, we cannot repeat "tomorrow" N times and verify whether it rains $0.7N$ times. Therefore, we cannot easily apply the verification procedure from the previous example in this scenario.

To resolve this issue, we can focus on *all* forecasts with a 70% confidence estimate of rain instead of focusing on a *single* forecast with this estimate. Suppose the meteorologists make N forecasts of rain with a confidence estimate of 70%, together with $0.76N$ actual observations of rain. Since it rained in 76% of the forecasts, a 70% confidence estimate might be deemed reliable (if N is sufficiently large). Whether or not this deviation of 6% is deemed acceptable depends on the task at hand. Whereas various tasks might accept such a minor deviation, others might not. In this forecasting task, we assume that a deviation of 6% is acceptable. Then, the meteorologists were able to accurately reflect the uncertainty about their predictions for this confidence estimate. However, suppose that we instead observed rain in 50% of these forecasts associated with a 70% confidence estimate. Then, if N is sufficiently large, they could be deemed unreliable as an observation of 50% deviates substantially from the estimated 70%. However, whether the deviation is acceptable seems to depend on the task at hand. Generally, no clear cut exists between reliable and unreliable confidence estimates. Therefore, rather than declaring a confidence estimate as either reliable or unreliable, we instead focus on its degree of reliability.

The degree of reliability is captured by the concept of *calibration*. Calibration measures the discrepancy between forecasts with a given confidence estimate and their empirical frequency. The confidence estimate is said to be *well-calibrated* (or

calibrated) if this discrepancy is small. Complementary, the confidence estimate is said to be *miscalibrated* (or poorly calibrated) if this discrepancy is large. We formalize this discrepancy mathematically in § 2.3.

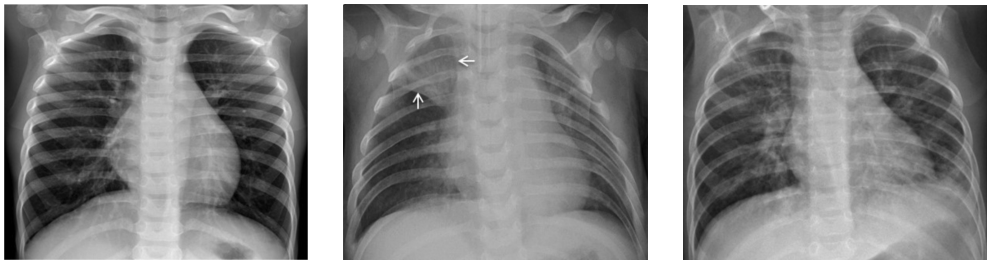
Although we mainly focused on the calibration of a single confidence estimate (0.7 in the example above), we would like all confidence estimates to be well-calibrated. If all confidence estimates given by a forecasting model are well-calibrated, then the forecasting model itself is called well-calibrated. Similarly, if most of its confidence estimates are miscalibrated, then the model itself is called miscalibrated. Well-calibrated confidence estimates are desirable as they are meaningful and intuitive. In contrast, miscalibrated confidence estimates can be considered meaningless and unintuitive as they fail to reflect reality. Moreover, miscalibrated confidence estimates can provide a false sense of security and mislead end-users to trust incorrect estimates. Therefore, good calibration is a desirable property for any forecasting model.

2.1.3 Classification tasks

The relevance of calibration is not limited to weather forecasting. Generally, calibration is useful in any prediction task. This includes classification tasks in which the goal is to classify a sample as one of the possible *classes* (or *categories*). In this work, we focus on *image classification tasks*, which means the sample for which we try to predict its class is an image. For instance, for an *image* of a dog, we could try to predict its breed. In this case, all possible dog breeds are the *classes*. Let us consider a more specific example of an image classification task that we will use as a running example throughout this chapter.

Pneumonia classification

According to the *World Health Organization* (WHO), pneumonia is the most prominent contagious cause of death in children worldwide [52]. According to the WHO, pneumonia accounts for roughly 15% of all deaths of children under five years old [52]. They further report that pneumonia affects children and families everywhere but is most prevalent in South Asia and Africa. The two leading causes of pneumonia are bacterial and viral pathogens that require substantially different forms of treatment [28, 43]. In particular, bacterial pneumonia demands immediate antibiotic therapy, while viral pneumonia demands supportive care. Hence, there is a compelling reason to distinguish between these two causes. Chest X-rays are an indispensable element of pneumonia diagnoses as they can assist in distinguishing between different types of pneumonia. However, distinguishing between different types of pneumonia is remarkably challenging,



(a) No pneumonia **(b)** Bacterial pneumonia **(c)** Viral pneumonia

Figure 2.1: Illustration of three chest X-ray images, from [18]. The chest X-ray without pneumonia (a) depicts clear lungs without any areas of abnormal opacification in the image. Bacterial pneumonia (b) generally exhibits a consolidation in one of the lung lobes (see the white arrows), whereas viral pneumonia (c) shows a more diffuse pattern in both lungs.

even when X-ray images are available. For instance, consider Figure 2.1 that illustrates the three possible scenarios concerning pneumonia, namely an X-ray that contains (1) no pneumonia, (2) bacterial pneumonia, and (3) viral pneumonia. Since these different scenarios are so challenging to distinguish, a domain expert is required. However, fast radiologic interpretation of X-rays is not always readily available, particularly not in low-resource environments. Therefore, there is a strong motive to automatically classify whether a patient has one of the types of pneumonia or not. Fortunately, such a system can be designed with machine learning.

In order to tackle pneumonia classification with machine learning, we first need to interpret the problem as a classification task. We would like to apply machine learning to decide whether a chest X-ray image is associated with either (1) no pneumonia, (2) bacterial pneumonia, or (3) viral pneumonia. In this case, the chest X-ray image represents the input and the three possible classes represent the output. The general objective of this task is to predict which class each input image belongs to. In the context of machine learning, such a prediction is performed using a *classifier* or *model*. In an image classification task, such a classifier maps an image to a probability distribution over classes. This probability distribution is represented by a vector consisting of a probability for each class. Each probability represents the classifier's confidence about an input being of that class. Therefore, we refer to this vector as the *confidence score vector*. This vector consists of probabilities and each probability represents a *confidence score*. The confidence score of each class takes a value between 0 and 1 and all confidence scores sum to 1. Typically, the predicted class of an

image is chosen as the class associated with the highest confidence score in the confidence score vector.

Example 2.1.1 (Prediction in the running example)

Given a particular chest X-ray image, a classifier might output a confidence score vector $(0.3, 0.6, 0.1)$. The confidence scores in this vector indicate the classifier's confidence about the "no pneumonia", "bacterial pneumonia", and "viral pneumonia" classes, respectively. Since the highest confidence is associated with the "bacterial pneumonia" class, the classifier will predict the image being of this class.

Calibration for pneumonia classification

Calibration is not only relevant for weather forecasting but also pneumonia classification. To exemplify this statement, consider the following scenario. In a hospital, only a limited number of doctors can detect (a type of) pneumonia from X-ray images. Due to this limitation, the doctors could employ a machine learning classifier that classifies whether a patient has a type of pneumonia or not. Suppose that the doctors further investigate patients based on their associated confidence score for bacterial pneumonia. If the doctors have too little time to go through all X-rays, they could focus on a pre-selection made by the classifier based on the predicted confidence scores for bacterial pneumonia. Then, the doctors would first further investigate X-rays of patients with the highest confidence scores for bacterial pneumonia. Suppose that the X-ray of a particular patient receives a confidence score of 60% for bacterial pneumonia. Furthermore, assume that this confidence score is associated with an empirical frequency of 95%. Based on the predicted confidence score of 60%, the doctor may decide to postpone further investigation of this patient and focus on one of their other (more urgent) tasks instead. However, this patient would have received (almost) immediate further investigation if the doctor had been aware of the empirical frequency of 95%. Therefore, the miscalibrated confidence score of 60% can negatively impact the patient. Hence, miscalibrated confidence scores can be misleading and unintuitive for decision-making processes. Consequently, calibration is an important, if not at least desirable, property in various classification tasks.

Next to their relevance in safety-critical applications, well-calibrated confidence scores also help establish trustworthiness with the user [9]. Trustworthiness is crucial for numerous machine learning classifiers, such as neural networks, since they act as black boxes whose internal reasoning is generally challenging

to understand by humans. The previously described reasons render calibration a desirable property for many classifiers.

2.2 Mathematical setting

In order to formalize calibration, we first need to introduce the mathematical context of this work. To this end, we start by formalizing classification tasks and classifiers. Throughout this work, we study and model relationships of *inputs* and corresponding *targets* for the purpose of predicting targets for unseen inputs. As highlighted previously, predicting the target of an unseen input based on the relationship between known inputs and corresponding targets is called *classification*. We also discussed that classification is performed with a classifier that maps inputs to targets. This work focuses on inputs represented as images and targets represented as items from a discrete set of class labels. Throughout this work, an image is denoted by X and a class label by Y.

In order to quantify calibration mathematically, we let image X and class label Y be random variables in some common probability space. Following the notation introduced by Vaicenavicius et al. [48], we consider N independent and identically distributed image-label pairs (X, Y) that are given by $(X^{(1)}, Y^{(1)}), \ldots, (X^{(N)}, Y^{(N)}) \in \mathcal{X} \times \mathcal{Y}$, all drawn from the same joint distribution $\mathbb{P}(X, Y)$. Here, $\mathcal{X}$ denotes the input space (*i.e.*, all possible images) and $\mathcal{Y} = \{1, 2, \ldots, K\}$ the label space, consisting of K different class labels.

Example 2.2.1 (Classes in the running example)

Consider the pneumonia classification task proposed in § 2.1.3. In this case, the label space is given by $\mathcal{Y} = \{1, 2, 3\}$, where class 1, 2, and 3 denote the "no pneumonia", "bacterial pneumonia", and "viral pneumonia" classes, respectively. If some chest X-ray image X contains viral pneumonia, its corresponding target would be given by $Y = 3$. However, if X contains no pneumonia, its corresponding target would be given by $Y = 1$.

Given a training set of realized outcomes $\{(x^{(j)}, y^{(j)})\}_{j=1}^{N}$ of $\{(X^{(j)}, Y^{(j)})\}_{j=1}^{N}$, the objective is to find a classifier that predicts the correct class label for each possible image in the input space. (This notation for realized outcomes will primarily be used for definitions based on finite sets of samples. We generally stick to the random variables in examples.) Ideally, we would like to find the optimal (probabilistic) classifier $\mathbf{g}^{\text{opt}} = \mathbb{P}(Y|X)$ [13, 48]. Unfortunately, it is typically unfeasible to find the optimal classifier given a training set of finitely many samples. Hence, we aim to to find a classifier that closely approximates

the optimal classifier instead. Formally, we would like to find a classifier $\mathbf{g} : \mathcal{X} \to \Delta^K$ with $\mathbf{g} \approx \mathbf{g}^{\text{opt}}$. Here, Δ^K denotes a space where each point represents a vector $\mathbf{q} = (q_1, q_2, \ldots, q_K)$ such that each $q_k \in [0, 1]$ and $\sum_k q_k = 1$. Formally, $\Delta^K = \{(q_1, q_2, \ldots, q_K) \in [0, 1]^K | \sum_{k=1}^K q_k = 1\}$ denotes the $(K-1)$-dimensional probability simplex. To ease readability, we typically use boldface characters such as $\mathbf{g}$, $\mathbf{q}$, and so forth, to denote elements of Δ^K or functions whose range is Δ^K. Classifier $\mathbf{g}$ can be constructed in various manners. For instance, it can correspond to a neural network, a random forest, or any one-vs-all binary classification model, such as logistic regression. In this work, we specifically focus on neural network classifiers. Therefore, throughout this work, the term "classifier" refers to a neural network classifier unless stated otherwise.

Remark 2.2.1 (Remark on classifiers parameters)

Typically, $\mathbf{g}$ is written as $\mathbf{g}_\theta$, where θ represents its parameters (*e.g.*, weights and biases). In this work, however, these parameters will be irrelevant, and hence, we simply write $\mathbf{g}$ to represent a classifier.

For a particular input sample X, classifier $\mathbf{g}$ produces a probability distribution over the classes $\mathbf{Z} = \mathbf{g}(X) \in \Delta^K$. As discussed in the previous section, we refer to this distribution as the *confidence score vector* since this distribution is a vector of confidence scores, one for each class. In particular, $\mathbf{Z} = (Z_1, Z_2, \ldots, Z_K)$ is a random vector consisting of a random variable Z_k for each class k. Each confidence score Z_k indicates the confidence that classifier $\mathbf{g}$ has about input X being of class k.

Example 2.2.2 (Notation in the running example)

Consider the classification task presented in § 2.1.3. In this case, the label space is given by $\mathcal{Y} = \{1, 2, 3\}$, where classes 1, 2, and 3 denote the "no pneumonia", "bacterial pneumonia", and "viral pneumonia" classes, respectively. Consider a classifier $\mathbf{g}$ that produces confidence score vector $\mathbf{Z} = (0.05, 0.1, 0.85)$ for a chest X-ray X. A confidence score of 0.85 for class 3 can be interpreted as classifier $\mathbf{g}$ being 85% confident about X-ray X being of class 3. Furthermore, the classifier predicts class 3 for input X since its associated confidence score is the highest across all scores.

Throughout the remainder of this work, we primarily disregard class label Y and instead use its one-hot encoded representation $\mathbf{Y} = (Y_1, Y_2, \ldots, Y_K)$, where $Y_k = 1$ if X is of class k and $Y_k = 0$ otherwise. Importantly, we refer to $\mathbf{Y}$ as the *ground truth vector*. A schematic view of a classifier $\mathbf{g}$, together with an image X, its corresponding ground truth vector $\mathbf{Y}$, and confidence score

Figure 2.2: Schematic overview of an input pair $(X, \mathbf{Y})$ together with a classifier $\mathbf{g}$ that predicts confidence score vector $\mathbf{Z}$ for input sample X.

vector $\mathbf{Z}$ is depicted in Figure 2.2. The classifier is represented as a black box to emphasize that we do not focus on the classifier itself but rather on evaluating its confidence scores.

2.3 Calibration formalization

For a well-calibrated classifier, the confidence score vector $\mathbf{Z}$ should be well-calibrated. Intuitively, this means that among all samples with confidence score vector $\mathbf{Z}$, the class distribution[1] $\mathbb{E}[\mathbf{Y}|\mathbf{Z}]$ should be (approximately) distributed as $\mathbf{Z}$. Throughout this work, we refer to this distribution as the *calibrated score vector* $\mathbf{C} = \mathbb{E}[\mathbf{Y}|\mathbf{Z}]$. This calibrated score vector can also be written as $\mathbf{C} = (C_1, C_2, \ldots, C_K)$, where calibrated score $C_k = \mathbb{E}[Y_k|\mathbf{Z}]$ [22]. Each calibrated score C_k indicates the proportion of ground truth class k among *all* samples for which the classifier predicted confidence score vector $\mathbf{Z}$. Then, a confidence score vector $\mathbf{Z}$ is perfectly calibrated if and only if $\mathbf{Z} = \mathbf{C}$. The difference between confidence score vector $\mathbf{Z}$ and calibrated score vector $\mathbf{C}$ then represents the *calibration error* or *degree of miscalibration*. Figure 2.3 provides a schematic overview that extends the overview in Figure 2.2 with calibrated score vector $\mathbf{C}$. In order to get more intuition behind calibrated score vectors, consider the following example.

Example 2.3.1 (Miscalibration in the running example)

Consider a confidence score vector $\mathbf{Z} = (0.3, 0.4, 0.3)$ generated by classifier $\mathbf{g}$. Suppose that classifier $\mathbf{g}$ produced 10 predictions with confidence score vector $\mathbf{Z}$ in total, where each prediction is associated with a different input

[1] Throughout this work, expectations (denoted by $\mathbb{E}$) are taken over the distribution $\mathbb{P}(X)$ unless otherwise specified.

sample. Furthermore, suppose that the ground truth class for 4 of these samples is "1" and for the other 6 it is "2". These statistics are summarized in Table 2.1. Then, we can calculate the calibrated score vector $\mathbf{C}$ as follows. We find that $C_1 = \mathbb{E}[Y_1|\mathbf{Z}] = 4/10 = 0.4$ since 4 out of 10 samples with confidence score $\mathbf{Z}$ are of class 1. Similarly, $C_2 = \mathbb{E}[Y_2|\mathbf{Z}] = 6/10 = 0.6$ since 6 out of 10 samples with confidence score $\mathbf{Z}$ are of class 2. Furthermore, $C_3 = 0.0$. Therefore, the calibrated score vector is given by $\mathbf{C} = (0.4, 0.6, 0.0)$. Although we cannot quantify the degree of miscalibration just yet, we observe a moderate discrepancy between confidence score vector $\mathbf{Z} = (0.3, 0.4, 0.3)$ and calibrated score vector $\mathbf{C} = (0.4, 0.6, 0.0)$. This discrepancy indicates a moderate degree of miscalibration.

Table 2.1: Summary of the confidence, ground truth, and calibrated score vector associated with Example 2.3.1.

Quantity	Representation	# of associated samples
Confidence score vector $\mathbf{Z}$	(0.3, 0.4, 0.3)	10
Ground truth vector $\mathbf{Y}$ (1)	(1.0, 0.0, 0.0)	4
Ground truth vector $\mathbf{Y}$ (2)	(0.0, 1.0, 0.0)	6
Calibrated score vector $\mathbf{C}$	(0.4, 0.6, 0.0)	10

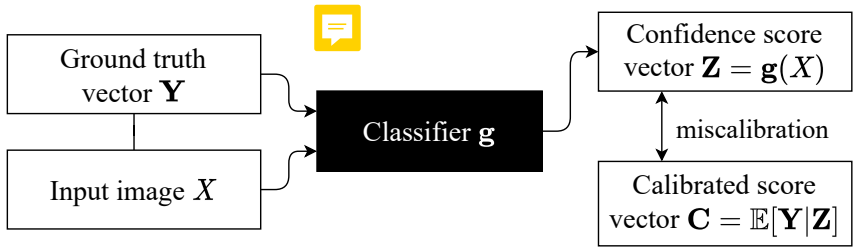


Figure 2.3: Schematic overview of an input pair $(X, \mathbf{Y})$ together with a classifier $\mathbf{g}$ that predicts confidence score vector $\mathbf{Z}$ for input sample X. Additionally, the calibrated score vector $\mathbf{C}$ is shown, together with the location where miscalibration manifests.

The concept of calibration we have been working with is often referred to as *multiclass calibration*. This notion of calibration requires the entire confidence score vector $\mathbf{Z}$ to be well-calibrated. However, measuring this notion of calibration is problematic, as we will discuss in § 2.3.2. Therefore, the literature primarily

focuses on two other notions of calibration. Before discussing these notions in more detail, let us first highlight the fundamental difference between the different notions of calibration. Generally, two relevant types of calibration exist. For a given confidence score vector $\mathbf{Z}$, we can require either (1) the entire vector $\mathbf{Z}$ to be well-calibrated or (2) the confidence score of the predicted class to be well-calibrated. This difference leads to different notions of calibration, some stronger than others. In this work, we consider two notions of calibration that require the entire vector $\mathbf{Z}$ to be well-calibrated. As discussed earlier, multi-class calibration is one of these notions. The other one is *class-wise calibration*. This notion of calibration requires each class to be well-calibrated. We highlight their difference in the following few sections. The other type of calibration only requires the confidence score of the predicted class, $\max_k Z_k$, to be well-calibrated. The notion of calibration that directly corresponds with this objective is called *confidence calibration*. A schematic overview of these different notions is illustrated in Figure 2.4. These notions of calibration are equivalent in binary classification tasks (*i.e.*, where the number of classes $K = 2$) but differ when the number of classes exceeds two.

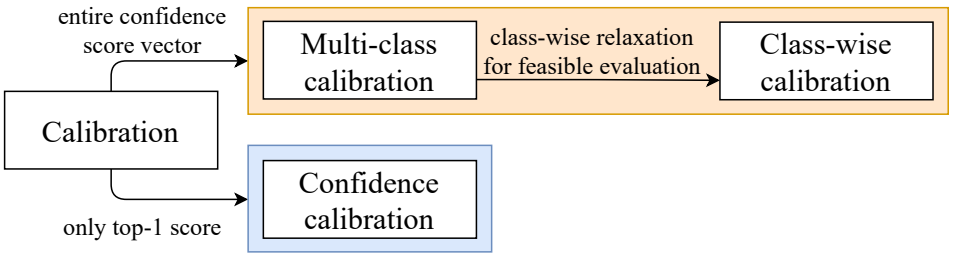


Figure 2.4: Schematic overview of the different notations of calibration.

2.3.1 Multi-class calibration

Multi-class calibration is the most stringent notion of calibration. This notion requires the entire confidence score vector $\mathbf{Z}$ to be well-calibrated. In other words, it requires confidence score vector $\mathbf{Z}$ to align with its calibrated score vector $\mathbf{C}$. Consider the following example for intuition behind this notion.

Example 2.3.2 (Simple multi-class calibration)

Consider the setting introduced in Example 2.3.1. In particular, we are given a confidence score vector $\mathbf{Z} = (0.3, 0.4, 0.2, 0.1)$ that leads to calibrated score vector $\mathbf{C} = (0.4, 0.6, 0.0, 0.0)$. For $\mathbf{Z}$ to be multi-class calibrated, it should

hold that $\mathbf{Z} = \mathbf{C}$. However, this is clearly not the case. Therefore, $\mathbf{Z}$ is not multi-class calibrated. Suppose, however, that the calibrated score vector would be given by $\mathbf{C} = (0.3, 0.4, 0.2, 0.1)$ instead. In this case, confidence score vector $\mathbf{Z}$ would be multi-class calibrated as $\mathbf{Z} = \mathbf{C}$.

In the example above, we find that multi-class calibration is remarkably stringent as no deviation between the confidence score and calibrated score vector is allowed. Formally, multi-class calibration is defined as follows.

Definition 2.3.1 (Multi-class calibration)
Consider a classifier $\mathbf{g} : \mathcal{X} \to \Delta^K$ with input X, corresponding ground truth vector $\mathbf{Y}$, and confidence score vector $\mathbf{Z} = \mathbf{g}(X)$. Classifier $\mathbf{g}$ is *multi-class calibrated* if the confidence score vector $\mathbf{Z}$ equals its calibrated score vector $\mathbf{C} = \mathbb{E}[\mathbf{Y}|\mathbf{Z}]$, *i.e.*, $\mathbf{Z} = \mathbf{C}$.

Unfortunately, the deviation of a confidence score vector from perfect calibration under multi-class calibration is generally too problematic to evaluate [11]. As further discussed in § 3.1.1, this deviation needs to be estimated using finitely many samples. Typically, the literature estimates this deviation using a binning procedure that groups confidence score vectors (and, consequently, their associated samples). However, the number of bins grows exponentially with the number of classes. Because of this and the limited number of samples, one has to estimate the calibrated score vectors. However, it is generally infeasible to come to an accurate estimation; see also Remark 4.3.1.

2.3.2 Class-wise calibration

Since multi-class calibration is generally too challenging to evaluate, most work in the literature focuses on *class-wise calibration* instead. Class-wise calibration is a relaxed version of multi-class calibration. For any confidence score Z_k, class-wise calibration requires that the proportion of class k among all samples with the same confidence score Z_k for class k equals Z_k [55]. Formally, class-wise calibration is defined as follows.

Definition 2.3.2 (Class-wise calibration [55])
Consider a classifier $\mathbf{g} : \mathcal{X} \to \Delta^K$ with input X, corresponding ground truth vector $\mathbf{Y}$, and confidence score vector $\mathbf{Z} = \mathbf{g}(X)$. Classifier $\mathbf{g}$ is *class-wise calibrated* if:

$$Z_k = \mathbb{E}[Y_k|Z_k] \quad \text{for all classes } k \in \mathcal{Y}.$$

The primary difference between multi-class calibration and class-wise calibration is that multi-class calibration requires confidence score Z_k to equal $C_k = \mathbb{E}[Y_k|\mathbf{Z}]$ for perfect calibration, whereas class-wise calibration requires confidence score Z_k to equal $\mathbb{E}[Y_k|Z_k]$ for perfect calibration. Hence, their difference resides in what is given in the conditional expectation. We provide a more detailed discussion on the difference between these two terms and when they are equivalent in § 4.3.2.

Notably, Remark 2.3.1 indicates that perfect calibration under multi-class calibration implies perfect calibration under class-wise calibration. Furthermore, Example 2.3.3 shows that perfect calibration under class-wise calibration need not necessarily imply perfect calibration under multi-class calibration. Therefore, class-wise calibration is, indeed, a weaker notion of calibration than multi-class calibration.

Remark 2.3.1 (**Multi-class calibration implies class-wise calibration, proven in Remark 4.3.2**)
If a classifier is perfectly calibrated under multi-class calibration, then it is also perfectly calibrated under class-wise calibration. Formally, if $\mathbb{E}[Y_k|\mathbf{Z}] = Z_k$ for every $k \in \mathcal{Y}$, then also $\mathbb{E}[Y_k|Z_k] = Z_k$.

Example 2.3.3 (Class-wise calibration does not necessarily imply multi-class calibration, modified from [48])
This example shows that perfect calibration under class-wise calibration does not necessarily imply perfect calibration under multi-class calibration. Consider the scenario of pneumonia classification with label space $\mathcal{Y} = \{1, 2, 3\}$, where class label 1 denotes "no pneumonia", class label 2 denotes "bacterial pneumonia", and class label 3 denotes "viral pneumonia". Suppose some classifier $\mathbf{g}$ exists that produces the confidence score vectors denoted in the first column of Table 2.2. The probability of these vectors occurring is equivalent, as indicated by the second column. Furthermore, the third column denotes the calibrated score vector for each confidence score vector.

Table 2.2: Information associated with the example, such as the confidence score vector outputted by classifier $\mathbf{g}$ and their calibrated score vectors.

$\mathbf{Z}$	$P(\mathbf{Z})$	$\mathbb{E}[\mathbf{Y}\|\mathbf{Z}]$
(0.1, 0.3, 0.6)	1/6	(0.2, 0.2, 0.6)
(0.1, 0.6, 0.3)	1/6	(0.0, 0.7, 0.3)
(0.3, 0.1, 0.6)	1/6	(0.2, 0.2, 0.6)
(0.3, 0.6, 0.1)	1/6	(0.4, 0.5, 0.1)
(0.6, 0.1, 0.3)	1/6	(0.7, 0.0, 0.3)
(0.6, 0.3, 0.1)	1/6	(0.5, 0.4, 0.1)

Let us first show that classifier $\mathbf{g}$ is perfectly calibrated under class-wise calibration. To this end, we first show that the first class is perfectly calibrated. In order to do so, we want to show that $\mathbb{E}[Y_1|Z_1] = Z_1$ for each unique value for Z_1. In this example, we have three unique values for Z_1, namely 0.1, 0.3, and 0.6. For $Z_1 = 0.1$, we find $\mathbb{E}[Y_1|Z_1 = 0.1] = (0.2 \cdot 0.5) + (0.0 \cdot 0.5) = 0.1 = Z_1$ since the calibrated scores of the first class are 0.2 and 0.0. For $Z_1 = 0.3$, we find $\mathbb{E}[Y_1|Z_1 = 0.3] = (0.2 \cdot 0.5) + (0.4 \cdot 0.5) = 0.3 = Z_1$. For $Z_1 = 0.6$, we find $\mathbb{E}[Y_1|Z_1 = 0.6] = (0.7 \cdot 0.5) + (0.5 \cdot 0.5) = 0.6 = Z_1$. Since all unique values for Z_1 are perfectly calibrated, we find that class 1 is perfectly calibrated. In a similar fashion, one can also show that class 2 and 3 are perfectly calibrated. Since all three classes are perfectly calibrated, classifier $\mathbf{g}$ is perfectly calibrated under class-wise calibration.

Let us now consider calibration under multi-class calibration. Consider the confidence score vector $(0.1, 0.3, 0.6)$ and its calibrated score vector $(0.2, 0.2, 0.6)$. Since a deviation exists between these vectors, we already find that classifier $\mathbf{g}$ is not perfectly calibrated under multi-class calibration. Therefore, a classifier can be perfectly calibrated under class-wise calibration while not being perfectly calibrated under multi-class calibration.

2.3.3 Confidence calibration

Confidence calibration is an entirely different notion of calibration compared to multi-class and class-wise calibration. Confidence calibration only requires the confidence score associated with the predicted class to be well-calibrated. Since the predicted class is chosen as the class with the highest confidence score in confidence score vector $\mathbf{Z}$, confidence calibration requires $\max_k Z_k$ to be well-calibrated. For more intuition behind this notion of calibration, consider the following example.

Example 2.3.4 (Example of confidence calibration)

Suppose we are given a confidence score vector $\mathbf{Z} = (0.80, 0.05, 0.15)$. Under confidence calibration, only the confidence score of the predicted class (*i.e.*, 0.80) should be well-calibrated. In order to verify this, we first consider all samples where the predicted class is associated with a confidence score of 0.80. Then, we test if the predicted class is correct for 80% of all these samples. In other words, when the confidence score of the predicted class is 0.80, the faction of samples where the predicted class is correct should also equal 0.80.

Generalizing the example above, confidence calibration can be described as follows. The confidence of the predicted class should equal the fraction of samples where the predicted class is correct. Hence, this notion of calibration aggregates all samples with a particular confidence score for the predicted class, regardless of their ground truth class. Formally, confidence calibration is defined as follows.

Definition 2.3.3 (Confidence calibration)

Consider a classifier $\mathbf{g} : \mathcal{X} \to \Delta^K$ with input X, corresponding ground truth vector $\mathbf{Y}$, and confidence score vector $\mathbf{Z} = \mathbf{g}(X)$. Let $q = \arg\max_k Z_k$ denote the predicted class of confidence score vector $\mathbf{Z}$. Classifier $\mathbf{g}$ is *confidence calibrated* if:

$$\mathbb{E}[Y_q | Z_q] = Z_q$$

assuming an arbitrary tie-breaking rule for arg max. Equivalently,

$$P(Y = \arg\max_k Z_k | \max_k Z_k) = \max_k Z_k. \tag{2.1}$$

At first sight, confidence calibration seems to capture the essence of calibration for many use cases. However, it can also introduce a risk if not all classes are equally important. This risk is illustrated in Example 2.3.5.

Example 2.3.5 (Insufficiency of confidence calibration, modified from [10])

Consider the scenario of pneumonia classification with label space $\mathcal{Y} = \{1, 2, 3\}$, where class label 1 denotes "no pneumonia", class label 2 denotes "bacterial pneumonia", and class label 3 denotes "viral pneumonia". Consider a perfectly calibrated classifier under confidence calibration that leads to the results denoted in Table 2.3.

Table 2.3: Information associated with the example. The first column denotes the confidence score vector and the second column denotes the probability of this confidence score vector occurring. The third column denotes the predicted class and the fourth column denotes the confidence score associated with the predicted class. Lastly, the fifth column denotes the actual probability of the predicted class given the confidence score vector.

$\mathbf{Z}$	$P(\mathbf{Z})$	$\arg\max_k Z_k$	$\max_k Z_k$	$P(Y = \arg\max_k Z_k \mid \mathbf{Z})$
(0.6, 0.3, 0.1)	0.50	1	0.6	0.3
(0.1, 0.6, 0.3)	0.25	2	0.6	0.9
(0.1, 0.3, 0.6)	0.25	3	0.6	0.9

We first verify that the top-1 confidence score of 0.6 is perfectly calibrated under confidence calibration:

$$\begin{aligned} P(Y = \arg\max_k Z_k \mid \max_k Z_k = 0.6) &= \big(0.50 \cdot P(Y = 1 \mid \mathbf{Z} = (0.6, 0.3, 0.1))\big) \\ &\quad + \big(0.25 \cdot P(Y = 2 \mid \mathbf{Z} = (0.1, 0.6, 0.3))\big) \\ &\quad + \big(0.25 \cdot P(Y = 3 \mid \mathbf{Z} = (0.1, 0.3, 0.6))\big) \\ &= \big(0.50 \cdot 0.3\big) + \big(0.25 \cdot (0.9 + 0.9)\big) \\ &= 0.15 + 0.45 \\ &= 0.6. \end{aligned}$$

In this example, predictions for either one of the three classes with a confidence score of 0.6 are perfectly calibrated under confidence calibration. For each class, however, the top-1 confidence score deviates significantly from the probability this class appears in reality for that score. If the predicted class is 1, then the top-1 confidence score of 0.6 is overconfident since $P(Y = 1 \mid \mathbf{Z} = (0.6, 0.3, 0.1)) = 0.3$. In contrast, if the predicted class is 2, then top-1 confidence score of 0.6 is underconfident since $P(Y = 2 \mid \mathbf{Z} = (0.1, 0.6, 0.3)) = 0.9$. Similarly, if the predicted class is 3, then top-1 confidence score of 0.6 is also underconfident since $P(Y = 3 \mid \mathbf{Z} = (0.1, 0.3, 0.6)) = 0.9$. Still, these predictions are perfectly calibrated under confidence calibration.

In this scenario, we would like patients that actually have either viral or bacterial pneumonia to be treated. Hence, it could be dangerous to classify patients that actually have a form of pneumonia as not having pneumonia. Moreover, classifying patients that actually have a form of pneumonia as not having pneumonia is worse than classifying patients that do not actu-

ally have pneumonia with either form of pneumonia. In other words, the pneumonia classes are more important than the no pneumonia class. In this example, we found that a top-1 confidence score of 0.6 is perfectly calibrated under confidence calibration. Based on this observation, one might believe that 60% of all patients associated with a confidence score vector of $(0.1, 0.6, 0.3)$ will actually have bacterial pneumonia. However, in reality, 90% of all patients associated with this vector will have bacterial pneumonia, as indicated by the fifth column in the table. Then, perfect calibration under confidence calibration can be somewhat misleading. Although the predictions are perfectly calibrated under confidence calibration, one could argue that they are not necessarily calibrated in a meaningful way. In order to avoid such an issue, one could require all classes to be well-calibrated instead of only the top-1 confidence score.

Chapter 3

Calibration metrics

In the previous chapter, we formally introduced the concept of calibration. We introduced three different notions of calibration. Given a classifier, each notion of calibration raises the following question: how can we measure the degree of miscalibration associated with the classifier? In this chapter, we address how to measure the degree of miscalibration for confidence and class-wise calibration. Furthermore, we introduce reliability diagrams as a visual tool to study miscalibration.

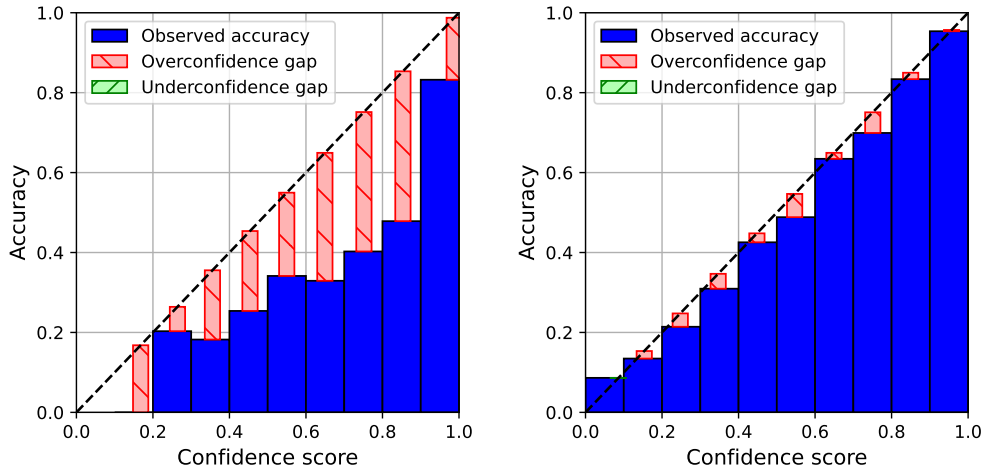
3.1 Reliability diagrams

Reliability diagrams provide a way to visually inspect calibration [5, 9, 34]. The literature introduces *confidence reliability diagrams* based on confidence calibration and *class-wise reliability diagrams* based on class-wise calibration. This section focuses on confidence reliability diagrams as class-wise reliability diagrams come with some issues that are listed in § 3.1.3.

A confidence reliability diagram allows for visual inspection of confidence calibration for a classifier based on a given set of predictions. For such a set of predictions, this diagram is constructed based on the confidence scores of the predicted classes. We also refer to the confidence score of the predicted class as the top-1 confidence score. Figure 3.1 illustrates two examples of confidence reliability diagrams. The horizontal axis of the diagram represents the top-1 confidence score, ranging from 0 to 1. The vertical axis of the diagram represents the empirical accuracy, also ranging from 0 to 1. The confidence reliability diagram then shows the empirical accuracy as a function of the top-1 confidence score.

3.1.1 Necessity of binning and estimation

In order to describe the construction of these diagrams, consider the existence of some classifier and a dataset of finite samples. The classifier predicts a confidence



(a) Overconfident **(b)** Better calibrated

Figure 3.1: Illustration of two reliability diagrams, each with ten bins of size 0.1 in the confidence score domain. Reliability diagram (a) shows overconfidence across all bins, whereas reliability diagram (b) seems much better calibrated.

score vector for each sample in the dataset. Due to the focus on confidence calibration, we concentrate on the top-1 confidence score predicted for each sample. The calibrated score for a specific top-1 confidence score can be estimated as the empirical accuracy across all predictions with that top-1 confidence score. For instance, the calibrated score for a top-1 confidence score of 0.98 equals the accuracy across all predictions with a top-1 confidence score of 0.98. Unfortunately, the top-1 confidence score can generally take many different values. (The precise number of unique values depends on the precision of the numbers used throughout computations.) Based on a limited number of predictions, it is likely that many, if not most, unique top-1 confidence scores only cover a few predictions. In this case, the calibrated score for each unique top-1 confidence score is computed as the accuracy based on only very few predictions, which is unlikely to lead to an accurate estimation. A solution to this problem is to group the top-1 confidence scores into a limited number of bins. For instance, confidence scores of 0.786 and 0.789 are almost equal; therefore, we would also expect their calibrated score to be roughly equal. Then, grouping predictions with distinct but roughly equivalent top-1 confidence scores seems reasonable. An example of such a binning scheme splits the top-1 confidence score domain $[0, 1]$ into ten equally-ranged bins of size 0.1. In this case, all predictions whose

top-1 confidence score falls in the $[0, 0.1)$ interval would be grouped into the first bin. Similarly, all predictions whose top-1 confidence score falls in the $[0.1, 0.2)$ interval would be grouped into the second bin, and so forth. After applying such a binning procedure, each bin contains zero or more predictions. The calibrated score for each bin is then estimated as the accuracy across all predictions in that bin. In contrast to earlier, the calibrated score is now computed for similar top-1 confidence scores instead of a single top-1 confidence score by considering all their predictions. In case a bin contains zero samples, the estimated accuracy is zero as well.

3.1.2 Constructing confidence reliability diagrams

Let us revisit confidence reliability diagrams and couple the previously introduced bins to these diagrams. Suppose we apply a binning procedure to a collection of top-1 confidence scores, resulting in ten bins. Then, a confidence reliability diagram plots each bin as a histogram, where the height of the histogram indicates the empirical accuracy in that bin. The reliability diagrams in Figure 3.1 visualize these histograms as blue bars. These diagrams indicate perfect calibration under confidence calibration with a black diagonal line. Therefore, a bin is perfectly calibrated if its blue bar aligns with the black diagonal line. In this case, the average accuracy in a bin equals its average confidence score. Similar to estimating the accuracy in a particular bin, the confidence score is estimated as its average in that bin. This quantity is particularly relevant when considering the smaller red and green bars in the reliability diagram, as these are centered on the average confidence score. These smaller bars are also called gaps, and they indicate the deviation between the estimated accuracy and estimated confidence. Therefore, they indicate the deviation from perfect calibration for each bin. In particular, red bars indicate that the estimated confidence is higher than the estimated accuracy. When the confidence is higher than the accuracy, we speak of overconfidence.

Example 3.1.1 (**Example of overconfidence in a bin**)

Consider a bin that captures all top-1 confidence scores in the $[0.6, 0.7)$ interval. Suppose a classifier provides predictions for five different input samples whose top-1 confidence scores are 0.6, 0.63, 0.65, 0.67, and 0.69. All top-1 confidence scores fall into the bin as they are contained in the $[0.6, 0.7)$ interval. Additionally, suppose the first four samples were predicted incorrectly while the last sample was predicted correctly. In this case, the estimated confidence in that bin equals $(0.60 + 0.63 + 0.65 + 0.67 + 0.69)/5 =$

0.648 and the estimated accuracy equals $(0 + 0 + 0 + 0 + 1)/5 = 0.2$. The estimated confidence of 0.648 is substantially higher than the estimated accuracy of 0.2. Intuitively, this discrepancy indicates that the classifier is, on average, more confident about these predictions than it should have been. In other words, the classifier is, on average, overconfident for the predictions that fall into this bin.

Similarly, green bars indicate that the estimated confidence is, on average, lower than the estimated accuracy. In this case, we speak of underconfidence. If most of the bins in a reliability diagram are overconfident, we tend to call the classifier overconfident. A similar principle holds for underconfidence. However, before calling a classifier overconfident or underconfident, we believe one should always examine the proportion of predictions in each bin. For instance, a classifier need not necessarily be underconfident if nine out of ten bins show underconfidence. If these nine bins only contain 5% of the total predictions, then the last bin contains the other 95%. If this last bin then shows overconfidence instead of underconfidence, one might reconsider calling the classifier underconfident. For that reason, we advocate for the practice of including a density diagram of the top-1 confidence scores with the reliability diagram. Accordingly, we also include a density diagram of the confidence score beneath the reliability diagram. For convenience, we also include vertical lines indicating the average accuracy and top-1 confidence score across all predictions. An example of these modified diagrams is illustrated in Figure 3.2. For this particular example, the density diagram of Figure 3.2a already indicates that the rightmost bin contains most predictions.

3.1.3 Class-wise reliability diagrams

Reliability diagrams for confidence calibration can be extended to class-wise calibration by plotting a reliability diagram for each class. In this case, the number of diagrams is equivalent to the number of classes. Each class-wise reliability diagram only includes the confidence scores associated with a single class. However, it becomes troublesome to compare how well-calibrated different classifiers are based on these diagrams alone. For instance, a classification task with 100 classes results in 100 class-wise reliability diagrams. In this case, it might be hard to compare different classifiers due to the large number of diagrams. Additionally, it is not always feasible to list this many diagrams in a paper. Due to these reasons, the literature primarily focuses on reliability diagrams for confidence calibration instead of class-wise calibration. Similarly, we will not ignore them in this work.

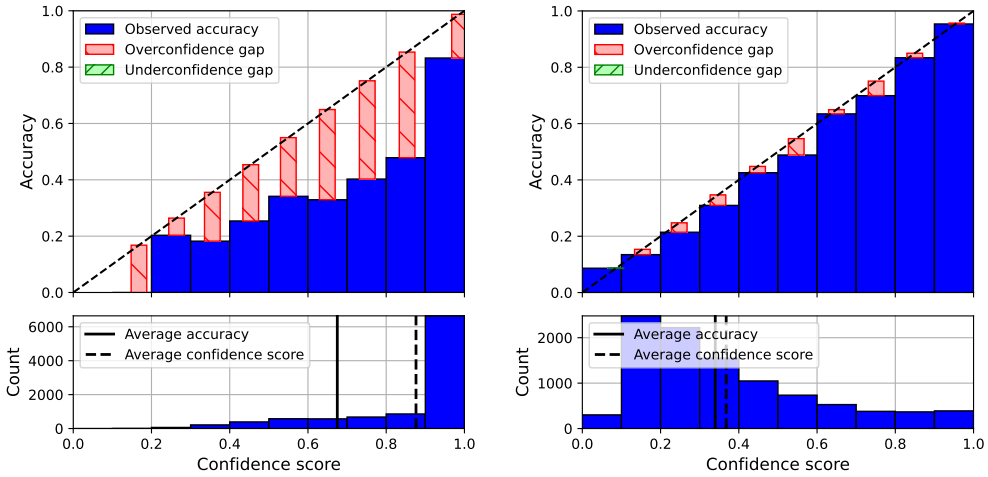


Figure 3.2: Illustration of two modified reliability diagrams that include density diagrams with the average accuracy and confidence score.

3.2 Expected calibration error

Although reliability diagrams are helpful visual tools for inspecting miscalibration, summary statistics for the degree of miscalibration (*i.e.*, a value that indicates the degree of miscalibration) might be more convenient in various use cases. For instance, summary statistics allow for a more straightforward comparison of different techniques that reduce the degree of miscalibration for a particular classifier. In the literature, a popular summary statistic for the degree of miscalibration is the *expected calibration error* (ECE). The ECE is an estimator of the degree of miscalibration. Since miscalibration has a different meaning for each notion of calibration, different variations of the ECE have been proposed. The *confidence-ECE* (abbreviated as conf-ECE in this work) estimates the degree of miscalibration for confidence calibration. Similarly, the *class-wise ECE* (CW-ECE) estimates the degree of miscalibration for class-wise calibration. Due to the reasons mentioned in § 2.3.2, no estimator for multi-class calibration has gained traction in the literature. Consequently, we exclude multi-class calibration estimation from our discussion. In the remainder of this section, we formalize the notion of confidence-ECE and class-wise ECE.

3.2.1 Confidence-ECE

Previously, we introduced confidence reliability diagrams that allow for visual inspection of miscalibration. The confidence-ECE is a summary statistic for the degree of miscalibration that is closely related to the confidence reliability diagram. This statistic is the average gap across all bins in the confidence reliability diagram, weighted by the number of samples in each bin. We formalize this notion throughout the remainder of this section. To this end, suppose we are given a set of N realized outcomes $\{(x^{(j)}, y^{(j)}\}_{j=1}^{N}$, together with their associated confidence score vectors $\{z^{(j)}\}_{j=1}^{N}$. In order to construct a confidence reliability diagram for these confidence scores, we need to select a binning scheme. Suppose we select a set of M disjoint bins $\{B_1, B_2, \ldots, B_M\}$, where each bin B_m contains all samples that fall into the confidence score interval given by I_m.

Example 3.2.1 (Example binning scheme)

For instance, an equally-ranged binning scheme of ten bins could be given by $B_1, B_2, \ldots, B_{10}$, together with confidence score intervals $I_1 = [0.0, 0.1), I_2 = [0.1, 0.2), \ldots, I_M = [0.9, 1.0]$. Here, bin B_1 contains all samples whose confidence score fall inside the interval $I_1 = [0.0, 0.1)$.

Formally, the confidence-ECE is given as the average absolute gap across all bins, weighted by the number of samples in each bin:

$$\text{conf-ECE} = \sum_{m=1}^{M} \frac{|B_m|}{N} \big|\text{acc}(B_m) - \text{conf}(B_m)\big|,$$

where $|B_m|$ the number of samples in bin B_m. Furthermore, $\text{acc}(B_m)$ and $\text{conf}(B_m)$ estimate the accuracy and confidence score in bin B_m. For bin B_m, $\text{acc}(B_m)$ and $\text{conf}(B_m)$ estimate the left-hand and right-hand side of Equation 2.1, respectively. Given a set of samples, the estimated accuracy $\text{acc}(B_m)$ is computed as the fraction of samples in bin B_m that were predicted correctly. Formally, this computation is given by:

$$\text{acc}(B_m) = \frac{1}{|B_m|} \sum_{j \in B_m} \mathbb{1}(\arg\max_k z_k^{(j)} = y^{(j)}),$$

where $z^{(j)}$ and $y^{(j)}$ denote the confidence score vector and ground truth class for sample j, respectively. Hence, $\arg\max_k z_k^{(j)}$ denotes the predicted class for sample j. Indicator function $\mathbb{1}$ equals 1 when its argument holds and 0, otherwise. Therefore, the indicator function returns 1 when the predicted class corresponds

with the ground truth class and 0 otherwise. In contrast, the estimated confidence score in bin B_m is computed as the average confidence score for the number one prediction. Formally, this computation is given by:

$$\text{conf}(B_m) = \frac{1}{|B_m|} \sum_{j \in B_m} \max_k z_k^{(j)}.$$

Clearly, the conf-ECE is minimized when $\text{acc}(B_m) = \text{conf}(B_m)$ for all $m \in \{1, \ldots, M\}$. In this case, the con-ECE reaches a value of 0. In contrast, its worst value is 1.

Another popular metric for confidence calibration based on the ECE is the squared variant of the confidence-ECE, often denoted as conf-ECE_2. This metric is given as:

$$\text{conf-ECE}_2 = \left(\sum_{m=1}^{M} \frac{|B_m|}{N} \left(\left| \text{acc}(B_m) - \text{conf}(B_m) \right|^2 \right) \right)^{\frac{1}{2}}.$$

Compared to the conf-ECE, this metric has increased sensitivity to poorly calibrated confidence scores. This metric might be preferred over the conf-ECE if poorly calibrated confidence scores are especially harmful. Since calibration is primarily of importance in safety-critical applications, this metric has been gaining more traction after its introduction [42].

For computing these metrics, Guo et al. [9] set the standard of using 15 bins of equal range in the confidence domain. Most work in the literature uses this standard (*e.g.*, see [25, 35, 42, 57]). The literature also employs this standard for other metrics of miscalibration, such as the one discussed in the subsequent section.

Example 3.2.2 (Example confidence-ECE values)

To get more intuition behind the values confidence-ECE can take, consider the confidence reliability diagrams illustrated in Figure 3.2. We derive their conf-ECE by averaging the gap across all bins and weighing each gap by the number of samples in each bin. Consequently, this leads to a conf-ECE of 0.201 for Figure 3.2a and a conf-ECE of 0.028 for Figure 3.2b. These values correspond with our intuition since Figure 3.2b seems much better calibrated than Figure 3.2a. Additionally, the conf-ECE_2 for Figure 3.2a is given by 0.217, whereas the conf-ECE_2 for Figure 3.2b is given by 0.031. These observations also correspond with our intuition.

3.2.2 Class-wise ECE

In contrast to the conf-ECE, the CW-ECE is a metric of class-wise calibration instead of confidence calibration [23]. As highlighted in the previous section, the conf-ECE employs a single binning scheme for the top-1 confidence scores across all predictions. The CW-ECE employs a single binning scheme for the confidence scores of each class k. Consequently, the CW-ECE leads to K binning schemes. The CW-ECE is then computed as the average gap across all class-wise bins, weighted by the number of samples in each bin. Instead of a single set of M bins, we now have a set of M bins $\{B_{k,1}, B_{k,2}, \ldots, B_{k,M}\}$ for each class k. For instance, bin $B_{7,3}$ represents the third bin when considering class 7. Formally, the CW-ECE is given as follows:

$$\text{CW-ECE} = \frac{1}{K} \sum_{k=1}^{K} \sum_{m=1}^{M} \frac{|B_{k,m}|}{N} \big|\text{prop}(B_{k,m}) - \text{conf}(B_{k,m})\big|,$$

where K denotes the number of classes, N denotes the total number of samples, M denotes the number of bins for each class, and $B_{k,m}$ denotes the mth bin for class k. Similarly to the conf-ECE, $\text{conf}(B_{k,m})$ denotes average confidence score in bin $B_{k,m}$. In contrast, the accuracy term is replaced by $\text{prop}(B_{k,m})$, which represents the actual proportion of class k in bin $B_{k,m}$. Similar to the conf-ECE_2, a squared variant can be defined for the CW-ECE. Typically, this variant is denoted as CW-ECE_2. Formally, CW-ECE_2 is given by:

$$\text{CW-ECE}_2 = \left(\frac{1}{K} \sum_{k=1}^{K} \sum_{m=1}^{M} \frac{|B_{k,m}|}{N} \Big(\big|\text{prop}(B_{k,m}) - \text{conf}(B_{k,m})\big|^2 \Big) \right)^{\frac{1}{2}}.$$

In contrast to CW-ECE, this metric increases sensitivity to poorly calibrated confidence scores. As argued in the previous section, this behavior might be favorable if poorly calibrated confidence scores tend to be especially harmful.

3.3 Miscalibration in practice

The calibration metrics proposed in this chapter allow for evaluating the degree of miscalibration of different classifiers. This section inspects the degree of miscalibration for four classifiers that stem from different popular neural network architectures. Specifically, we focus on LeNet-5, ResNet-110, ResNet-110 with Stochastic Depth, and DenseNet-40, all trained with the log-loss and light data augmentation on CIFAR-100. Details surrounding these classifiers, their training procedures, and data augmentation are given in Appendix A.

In order to evaluate the calibration of these classifiers, we focus on their confidence reliability diagrams. The focus on these diagrams stems from them being straightforward and easily interpretable compared to class-wise reliability diagrams and summary statistics. To exemplify the problem of class-wise reliability diagrams, consider evaluating a classifier on CIFAR-100 with class-wise reliability diagrams. This evaluation would lead to 100 different class-wise reliability diagrams, one for each class. This vast number of diagrams hampers interpretability. The challenge of summary statistics is rooted in the interpretation of their values. Specifically, the summary statistics lack a meaningful unit or scale, which renders their interpretation more difficult. Therefore, we find that reliability diagrams are especially useful for gaining intuition to what degree a classifier is miscalibrated. In contrast, summary statistics for the degree of miscalibration, such as the conf-ECE and CW-ECE, are more suitable for succinctly summarizing the effectiveness of different techniques that reduce the degree of miscalibration. Since we are interested in the degree of miscalibration of these classifiers, we focus on their confidence reliability diagrams.

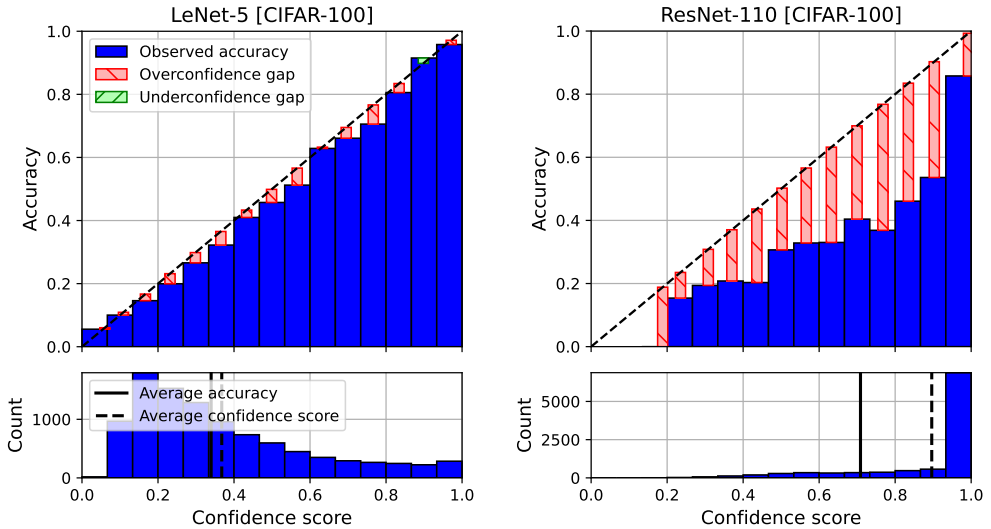
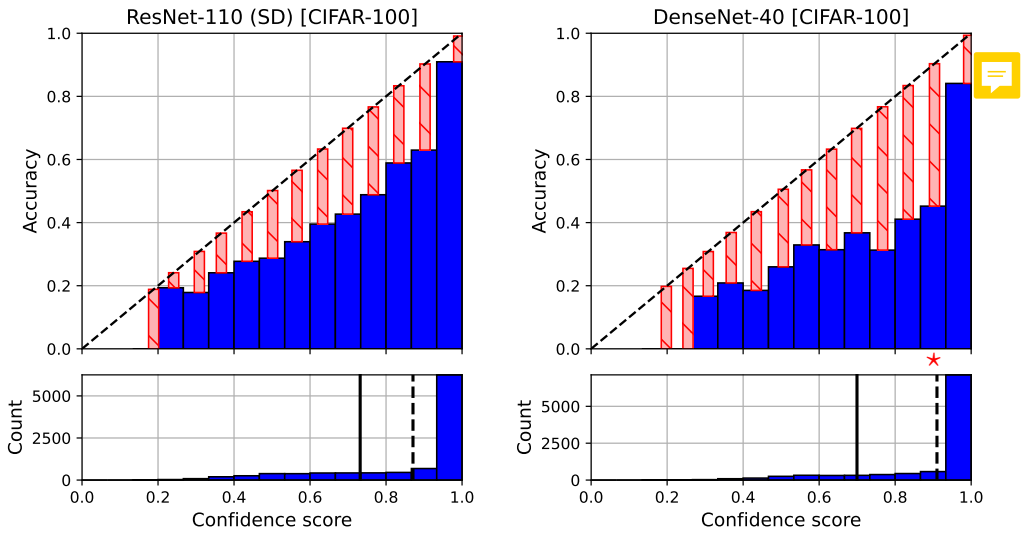


Figure 3.3: Illustration of confidence reliability diagrams consisting of 15 bins of equal range for (a) LeNet-5 and (b) ResNet-110 on CIFAR-100, trained using the log-loss.

Figure 3.3 and Figure 3.4 illustrate confidence reliability diagrams for the four classifiers on CIFAR-100. For all classifiers, we find at least some degree of



(a) **(b)**

Figure 3.4: Illustration of confidence reliability diagrams consisting of 15 equally-ranged bins for (a) Resnet-110 with Stochastic Depth and (b) DenseNet-40 on CIFAR-100, trained using the log-loss.

overconfidence. LeNet-5 (Figure 3.3a) outperforms any other other classifier substantially regarding confidence calibration. Notably, LeNet-5 has relatively low complexity (*e.g.*, number of parameters) while the other classifiers have substantially higher complexity. Let us exemplify how severe the degree of miscalibration can be for one of these more complex classifiers. For instance, consider the 14th bin in Figure 3.4b (see the red star). This bin contains all predictions whose top-1 confidence score lies in the $[0.866, 0.933)$ interval. For this bin, we observe that the average confidence lies around 0.9 (indicated by the location of the red bar), whereas the average accuracy in this bin is roughly 0.45. Therefore, on average, the classifier is twice as confident about these predictions as it would be under perfect confidence calibration. Hence, on average, the confidence scores of the predictions in this bin seem to possess a substantial degree of overconfidence. We do similar observations in many other bins across all complex classifiers (*i.e.*, those excluding LeNet-5). Hence, the complex classifiers seem to leverage overconfident predictions. We also evaluate the calibration of these classifiers on CIFAR-10 in Appendix C.1. This appendix also illustrates that these classifiers, excluding LeNet-5, tend to be overconfident in their predictions. Additionally,

this appendix shows that miscalibration appears regardless of the number of bins chosen for the reliability diagram.

To further support the finding that the complex classifiers are miscalibrated more severely, we list their conf-ECE$_2$ and CW-ECE$_2$ in Table 3.1. Then, we can compare their values to those of LeNet-5, which seemed relatively well-calibrated based on the confidence reliability diagram. We focus on the conf-ECE$_2$ and CW-ECE$_2$ instead of the conf-ECE and CW-ECE due to an interesting relationship we highlight in the following chapter (see § 4.4.2). We use 15 equally-ranged bins for computing the conf-ECE$_2$ and CW-ECE$_2$, as this is the standard in the literature (see the details in § 3.2.1). As already indicated by the confidence reliability diagrams, LeNet-5 outperformed all other classifiers concerning confidence calibration. This observation is also visible when inspecting the conf-ECE$_2$ in Table 3.1. The CW-ECE$_2$ values also indicate that LeNet-5 performs better concerning class-wise calibration. In other words, the more complex classifiers show substantially higher degrees of miscalibration across different calibration evaluation metrics in this experiment. In this section, we observed that neural network classifiers, especially the more complex ones, tend to be miscalibrated. Since miscalibration seems so prevalent, we would like to find out where it stems from. That is, we want to find possible causes of miscalibration.

Table 3.1: Comparison of four classifiers on CIFAR-100 concerning their degree of miscalibration. Upwards arrows indicate that higher values are better and downwards arrows indicate that lower values are better.

Classifier	**Accuracy (↑)**	**conf-ECE$_2$ (↓)**	**CW-ECE$_2$ (↓)**
LeNet-5	0.3398	0.0315	0.0164
ResNet-110	0.7088	0.2078	0.0272
ResNet-110 (SD)	0.7319	0.1602	0.0233
DenseNet-40	0.6996	0.2349	0.0296

Remark 3.3.1 (Unfair comparisons)

We would like to remark that classifiers cannot directly be compared based on their values for the calibration metrics introduced in this work. For instance, a classifier that attains an accuracy of 100% on a set of samples generally has an easier time performing well concerning calibration than a classifier that only attains an accuracy of 60% on the same set of samples. The classifier with an accuracy of 100% is perfectly calibrated if it predicts confidence score vectors where the confidence score of the predicted class is

1 and the confidence score of all other classes is 0. In contrast, the classifier that only attains an accuracy of 60% has a more challenging time reaching perfect calibration due to the variety of correctly and incorrectly predicted samples.

Chapter 4

Origination of miscalibration

In the previous chapter, we discussed several techniques for evaluating the degree of miscalibration in classifiers. We found that neural network classifiers, particularly more complex ones, tend to be miscalibrated. In this chapter, the objective is to find possible causes of miscalibration. In order to find these causes, we go through a multi-phase process. A schematic overview of this process, including all core sections of this chapter, is illustrated in Figure 4.1. We first introduce the objectives we would like a classifier to meet. Afterward, we dive into what a loss function of a classifier optimizes during training. Then, we relate what is optimized during training to the objectives we would like a classifier to meet. Based on their relation, we ultimately derive two hypotheses for the origination of miscalibration that we verify empirically with various experiments.

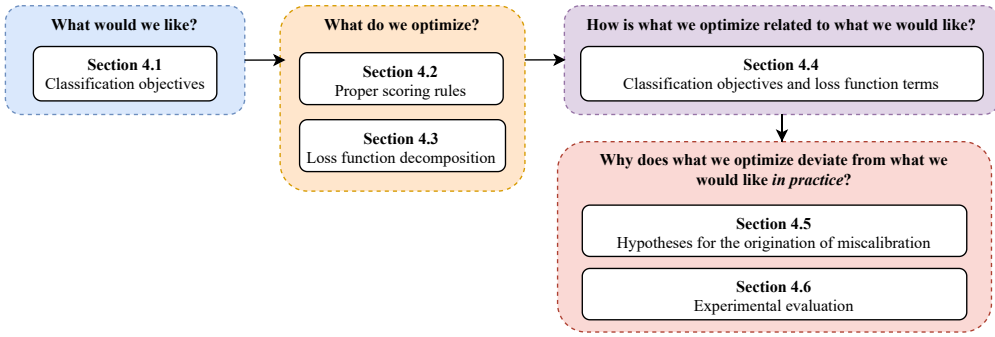


Figure 4.1: A schematic overview of the structure of Chapter 4.

4.1 Classification objectives

As highlighted in Figure 4.1, we start by introducing the qualities we would like a classifier to possess. Specifically, by acknowledging the relevance of calibration,

we argue that any classification task consists of two objectives. Afterward, we discuss that each objective can be viewed as a classification problem.

4.1.1 Objectives

For a particular input sample, the objective of a classifier is to predict the correct class, *i.e.*, the ground truth class. The predicted class is generally defined as the class with the highest score in the confidence score vector produced by the classifier. We can extend the objective for a single sample to an entire collection of samples by requiring the classifier to predict the correct class for each sample. Throughout this work, we refer to this objective as the *correctness objective*. In case the classifier leverages incorrect predictions, we speak of a classification error. Formally, the correctness objective aims to minimize the expected classification error, given by $\mathbb{E}[\mathbb{1}(\arg\max_k Z_k \neq \arg\max_k Y_k)]$. In other words, this objective aims to minimize the number of times where the predicted class deviates from the ground truth class.

The correctness objective alone might seem satisfactory as the primary idea of tackling a classification task is to provide correct predictions. However, as hinted in previous chapters, calibration is also a desirable property in many use cases. Throughout this chapter, we focus on multi-class calibration unless otherwise stated. With calibration in mind, a natural second objective arises. We refer to the desirability of calibration as the *calibration objective*. This objective aims to minimize the calibration error under multi-class calibration. The literature often defines this multi-class calibration error as the ℓ_2 norm difference between confidence score vector $\mathbf{Z}$ and calibrated score vector $\mathbf{C}$ (*e.g.*, see [14, 17, 24, 42]). This definition of the multi-class calibration error, denoted by MC-CE$_2$, is given by:

$$\text{MC-CE}_2 = \left(\mathbb{E}\left[|\mathbf{Z} - \mathbf{C}|^2\right]\right)^{\frac{1}{2}}. \tag{4.1}$$

Similar to the conf-ECE$_2$ and CW-ECE$_2$, the MC-CE$_2$ is more sensitive to poorly calibrated confidence scores, which tend to be more dangerous in applications [42].

4.1.2 Different objectives

By acknowledging the relevance of calibration, a classification task can be split into two objectives. On the one hand, the correctness objective ensures correct predictions. Therefore, this objective is directly related to optimizing the classification error (and accuracy). On the other hand, the calibration objective

ensures that the confidence score vectors correspond to their calibrated score vectors. These two objectives might seem somewhat related but are quite distinct. Specifically, a classifier that achieves a low accuracy is not necessarily miscalibrated and vice versa.

Example 4.1.1 (Comparing classification objectives)

Consider a classifier with an accuracy of 100% that always predicts the correct class with a confidence score of 70%. In this case, the *correctness objective* is fully met as the classifier always predicts the correct class label. However, the *calibration objective* is not met as the confidence score of 70% does not translate to the classifier predicting the correct class in 70% of all cases. Instead, the classifier always predicts the correct class label. Therefore, under perfect confidence calibration, its confidence score should be 100% as well.

4.1.3 Classification problems

Both objectives introduced in the previous section can be formulated as classification problems. While interpreting these objectives as classification problems might seem redundant, later sections highlight why this interpretation is helpful. For the correctness objective, the classifier has to predict the correct class out of many possible classes. Therefore, this objective can be viewed as a classification problem.

Contrary to the correctness objective, interpreting the calibration objective as a classification problem is non-trivial. The calibration objective focuses on ensuring that the confidence score vector aligns with its calibrated score vector. Fundamentally, this objective is a regression problem as the scores in the confidence score vector are continuous values. However, as discussed in § 3.1.1, confidences scores should be grouped to obtain meaningful calibrated scores. In § 3.1.1, we focused on the top-1 confidence scores as we were interested in confidence calibration. However, the calibration objective focuses on the entire confidence score vector. We intuitively expect each unique confidence score vector to capture even fewer samples than each unique top-1 confidence score as the vector consists of multiple confidence scores. Therefore, entire confidence score vectors should be grouped to obtain a meaningful calibrated score vector. Hence, this grouping procedure involves mapping a confidence score vector to one of the various groups, where each group computes its own calibrated score vector. This mapping can be viewed as a classification problem. Note, however, that this interpretation of the calibration objective as a classification problem deviates slightly from the typical notion of a classification problem.

Remark 4.1.1 (Deviation from common classification problems)
In a typical classification problem, a classifier predicts a sample to be one of several classes. Such a classification is either correct or incorrect. Therefore, the correctness of a prediction for a typical classification problem is easily verifiable by comparison with the ground truth class. In contrast, verifying whether the confidence score vector predicted for a specific sample is well-calibrated is more complicated. Specifically, whether the confidence score vector predicted for this sample is well-calibrated cannot be determined based on this sample alone. Instead, various other samples with similar confidence score vectors are required to get a meaningful estimate of a calibrated score vector. In contrast to a typical classification problem, the classification problem associated with the calibration objective requires multiple samples instead of a single sample.

4.2 Proper scoring rules

In the previous section, we argued that any classification task consists of a correctness and calibration objective. These are the two objectives that we would like to be optimized for a classifier. In the following two sections, we move to the next phase in Figure 4.1 and investigate what is actually optimized in the training procedure of a classifier. Specifically, this section introduces the concepts necessary to understand what is optimized during training. Since the remainder of this section will be mathematically involved, we first provide a straightforward textual explanation of the fundamental concepts. These fundamental concepts will be necessary to understand the remainder of this work. After this explanation, we formalize these concepts mathematically.

4.2.1 Informal discussion

The training of neural network classifiers consists of an optimization procedure, such as stochastic gradient descent. As part of the optimization procedure, the classifier's loss is calculated repeatedly for each step in the procedure. Calculating this loss requires the choice of a *loss function*. Since loss functions are generally designed for single samples, the loss on a set of samples is typically calculated by averaging the sample-wise losses. Afterward, the averaged sample-wise loss is used to update the classifier (*e.g.*, its weights) with the aim of the updated classifier achieving a lower loss.

When selecting a loss function, it is natural to require that the loss is minimized when a classifier provides the best possible predictions. As highlighted

in § 2.2, the best possible (probabilistic) classifier outputs the class conditional distribution $\mathbb{P}(\mathbf{Y}|X)$ for any sample X (sometimes also referred to as the true posterior class probabilities) [13, 48]. Loss functions whose loss is minimized when the classifier provides these predictions are called *proper scoring rules* [8, 29]. Therefore, proper scoring rules are promising and natural candidates for loss functions. Two popular proper scoring rules that are typically used as loss functions are the *log-loss* and *Brier score* (also known as the *cross-entropy* and *squared error*, respectively).

Proper scoring rules are not merely attractive loss functions. In particular, they also lead to a class of statistical divergences called *score divergences*. These divergences quantify the difference between two probability distributions. Such quantification can be particularly useful when examining to what degree a predicted confidence score vector resembles its calibrated score vector. Consequently, score divergences provide a manner to measure the degree of miscalibration, which highlights their importance in this work.

4.2.2 Formal discussion

Various concepts have been introduced informally in the previous section. This section aims to not only further elaborate on each concept but also to formalize them. Although these formalizations are not strictly necessary to understand for the remainder of the work, they provide much more intuition behind all the underlying relationships. Since mathematical definitions can be challenging to understand intuitively, we exemplify them with a running example. To this end, we let $\mathbf{p} = (p_1, p_2, \ldots, p_K) \in \Delta^K$ denote a confidence score vector generated by a classifier for some input sample x. Input sample x is also associated with a one-hot encoded ground truth vector $\mathbf{y} = (y_1, y_2, \ldots, y_K) \in \{0,1\}^K$ where $y_k = 1$ if x is of class k and $y_k = 0$ otherwise. Let us now start formalizing the concepts introduced previously using these two vectors.

Scoring rules As argued previously, proper scoring rules are natural and promising candidates for loss functions. Before we formally introduce proper scoring rules, we first introduce *scoring rules*. Afterward, we describe what it means for a scoring rule to be "proper".

Definition 4.2.1 (Scoring rule [31])
A *scoring rule* is a function $\phi : \Delta^K \times \{0,1\}^K \to \mathbb{R} \cup \{-\infty, +\infty\}$ that measures the quality of match between a confidence score vector $\mathbf{p} \in \Delta^K$ and its associated one-hot encoded ground truth vector $\mathbf{y} \in \{0,1\}^K$. The corresponding *score* resulting from a *scoring rule* is denoted $\phi(\mathbf{p}, \mathbf{y})$.

Throughout this work, we consider negatively-oriented scoring rules (*i.e.*, lower values are better). Therefore, we often simply refer to the *score* as the *loss*. Two well-known scoring rules are the *log-loss* ϕ^{LL} and *Brier score* ϕ^{BS}, both of which are the focus of this work. These two rules are commonly used as loss functions for optimizing machine learning classifiers. Table 4.1 describes their details, where by slight abuse of notation, p_y denotes the probability of the ground truth class (*i.e.*, $p_y = p_k$ for k such that $y_k = 1$).

Table 4.1: Scoring rules used throughout this work.

Name	Definition
Log-loss	$\phi^{LL}(\mathbf{p}, \mathbf{y}) = -\log p_y$
Brier score[1]	$\phi^{BS}(\mathbf{p}, \mathbf{y}) = \sum_{k=1}^{K}(p_k - y_k)^2$

Example 4.2.1 (Example calculation for a scoring rule)

Suppose a classifier is given some sample x for which it predicts a confidence score vector $\mathbf{p} = (0.4, 0.3, 0.2, 0.1)$. Additionally, suppose that the ground truth class of sample x is class 0. Therefore, the associated ground truth vector is given by $\mathbf{y} = (1, 0, 0, 0)$. If we were to evaluate the quality of this prediction with the Brier score ϕ^{BS}, we would obtain a loss of:

$$\begin{aligned}\phi^{BS}(\mathbf{p}, \mathbf{y}) &= \sum_{k=1}^{K}(p_k - y_k)^2 \\ &= (0.4 - 1)^2 + (0.3 - 0)^2 + (0.2 - 0)^2 + (0.1 - 0)^2 \\ &= 0.36 + 0.09 + 0.04 + 0.01 \\ &= 0.5.\end{aligned}$$

One might wonder how good this score is. Note that the best possible Brier score is 0 while the worst possible Brier score is 2. Whether a score of 0.5 is satisfactory depends on the application. The best possible prediction is obtained when confidence score vector $\mathbf{p}$ equals ground truth class vector $\mathbf{y}$, *i.e.*, when $\mathbf{p} = (1, 0, 0, 0)$.

Proper scoring rules Before defining what it means for a scoring rule to be "proper", we first have to introduce the *expected score* of a scoring rule. Notably,

[1] This definition of the Brier score is a natural multi-class extension of the original definition by Brier [3] (see, *e.g.*, [22]).

the expected score will only be used for defining proper scoring rules and score divergences throughout the remainder of this work. Hence, it is not of primary importance for the remainder of the work.

Previously, we discussed that a confidence score vector $\mathbf{p}$ is assigned a score $\phi(\mathbf{p}, \mathbf{y})$ if the true outcome is ground truth vector $\mathbf{y}$. Suppose that $\mathbf{y}$ is being sampled from a probability distribution $\mathbf{q} \in \Delta^K$ over K classes. In this case, $s(\mathbf{p}, \mathbf{q})$ denotes the expected score of confidence score vector $\mathbf{p}$ under distribution $\mathbf{q}$. If we draw output classes according to the distribution given by $\mathbf{q}$ and the classifier always returns confidence score vector $\mathbf{p}$, then $s(\mathbf{p}, \mathbf{q})$ gives the average score. In practical settings, confidence score vector $\mathbf{p}$ denotes the predicted confidence score vector produced by a classifier for a particular input sample. Given this confidence score vector, vector $\mathbf{q}$ typically denotes its associated calibrated score vector. Formally, the expected score is defined as follows.

Definition 4.2.2 (Expected score [22])
Let $\mathbf{p} \in \Delta^K$ denote a confidence score vector and $\mathbf{q} \in \Delta^K$ a probability distribution over K classes. The *expected score* of scoring rule ϕ on confidence score vector $\mathbf{p}$ with respect to class labels randomly drawn from $\mathbf{q}$ is defined as follows:

$$s(\mathbf{p}, \mathbf{q}) = \mathbb{E}_{\mathbf{Q} \sim \mathbf{q}}[\phi(\mathbf{p}, \mathbf{Q})] = \sum_{k=1}^{K} \phi(\mathbf{p}, \mathbf{e}_k) q_k, \tag{4.2}$$

where $\mathbf{Q}$ denotes a one-hot encoded vector randomly drawn from $\mathbf{q}$ and $\mathbf{e}_k$ denotes the one-hot encoded vector with ground truth class k (*i.e.*, $\mathbf{e}_k$ is a vector of length K with a 1 at position k and 0 everywhere else).

An example calculation of the expected score is given in Example 4.2.2.

Example 4.2.2 (Example calculation for the expected score)
Suppose a classifier is given some sample x for which it predicts a confidence score vector $\mathbf{p} = (0.4, 0.3, 0.2, 0.1)$. Furthermore, suppose that the calibrated score vector is represented by $\mathbf{q} = (0.5, 0.2, 0.1, 0)$. Note that we also use iden-

tity vectors $(\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3, \mathbf{e}_4) = ((1,0,0,0),(0,1,0,0),(0,0,1,0),(0,0,0,1))$. The expected score $s(\mathbf{p}, \mathbf{q})$ of scoring rule ϕ^{BS} is given by:

$$\begin{aligned} s(\mathbf{p}, \mathbf{q}) &= \sum_{k=1}^{K} \phi^{BS}(\mathbf{p}, \mathbf{e}_k) q_k \\ &= \left[(0.4-1)^2 + (0.3-0)^2 + (0.2-0)^2 + (0.1-0)^2\right] \cdot 0.5 \\ &\quad + \left[(0.4-0)^2 + (0.3-1)^2 + (0.2-0)^2 + (0.1-0)^2\right] \cdot 0.2 \\ &\quad + \left[(0.4-0)^2 + (0.3-0)^2 + (0.2-1)^2 + (0.1-0)^2\right] \cdot 0.1 \\ &\quad + \left[(0.4-0)^2 + (0.3-0)^2 + (0.2-0)^2 + (0.1-1)^2\right] \cdot 0.0 \\ &= 0.25 + 0.14 + 0.09 + 0 \\ &= 0.48. \end{aligned}$$

Similar to Example 4.2.1, the quality of this prediction depends on the specific use case. There is, however, still substantial room for improvement as the minimum value is zero.

With the notion of the expected score, we can define what it means for a scoring rule to be "proper". A scoring rule ϕ is said to be *proper* if its expected score s is minimized when predicting distribution $\mathbf{q}$ as confidence score vector $\mathbf{p}$. Formally, this statement is defined as follows.

Definition 4.2.3 (Proper scoring rules [13])

A scoring rule ϕ is called *proper* if for any $\mathbf{p}, \mathbf{q} \in \Delta^K$, the following inequality holds:

$$s(\mathbf{q}, \mathbf{q}) \leq s(\mathbf{p}, \mathbf{q}). \tag{4.3}$$

The scoring rule is called *strictly proper* if the inequality is strict for all $\mathbf{p} \neq \mathbf{q}$.

This definition indicates that the expected score of a proper scoring rule is minimized if $\mathbf{p} = \mathbf{q}$. Additionally, it highlights that the expected score of a *strictly* proper scoring rule is minimized if and only if $\mathbf{p} = \mathbf{q}$. Both the log-loss ϕ^{LL} and Brier score ϕ^{BS} are examples of strictly proper scoring rules [22]. For more details on proper scoring rules and expected scores, we refer to the work of Gneiting and Raftery [8].

Score divergences As argued previously, proper scoring rules are not merely attractive loss functions. In particular, a class of statistical divergences arises

from scoring rules. A statistical divergence is a type of measure that quantifies the dissimilarity between two probability distributions [46]. Throughout this work, we refer to the divergences that arise from scoring rules as *score divergences*. Based on a specific proper scoring rule, these divergences quantify the difference between two probability distributions, such as the confidence score vector $\mathbf{p}$ and ground truth vector $\mathbf{y}$. Formally, the score divergence is defined as follows.

Definition 4.2.4 (Score divergences [22])
Let ϕ be a proper scoring rule. Then, *score divergence* $d : \Delta^K \times \Delta^K \to \mathbb{R}_{\geq 0}$ under proper scoring rule ϕ is defined as:

$$d(\mathbf{p}, \mathbf{q}) = s(\mathbf{p}, \mathbf{q}) - s(\mathbf{q}, \mathbf{q}) \tag{4.4}$$

for all $\mathbf{p}, \mathbf{q} \in \Delta^K$.

Weijs et al. [49] show that the score divergence of the Brier score ϕ^{BS} is the squared loss $d^{BS}(\mathbf{p}, \mathbf{q}) = \sum_{k=1}^{K}(p_k - q_k)^2$ and that the score divergence of the log-loss ϕ^{LL} is the KL-divergence $d^{LL}(\mathbf{p}, \mathbf{q}) = \sum_{k=1}^{K} q_k \log \frac{q_k}{p_k}$. Table 4.2 summarizes these score divergences. Notably, from the definition of the score

Table 4.2: Score divergences used throughout this work.

Name	Definition
Log-loss	$d^{LL}(\mathbf{p}, \mathbf{q}) = \sum_{k=1}^{K} q_k \log \frac{q_k}{p_k}$
Brier score	$d^{BS}(\mathbf{p}, \mathbf{q}) = \sum_{k=1}^{K}(p_k - q_k)^2$

divergence, one can easily verify that scoring rules are proper if and only if their associated score divergence is always non-negative. Throughout the remainder of this work, we primarily employ score divergences for measuring the degree of miscalibration between a confidence score vector and its associated calibrated score vector.

Example 4.2.3 (Example calculation for a score divergence)
To exemplify score divergences, consider computing the score divergence under the Brier score ϕ^{BS} given confidence score vector $\mathbf{p} = (0.4, 0.3, 0.2, 0.1)$ and ground truth vector $\mathbf{y} = (1, 0, 0, 0)$. Its associated score divergence can be calculated in two manners, namely (1) $d(\mathbf{p}, \mathbf{y}) = s(\mathbf{p}, \mathbf{y}) - s(\mathbf{y}, \mathbf{y})$ and (2) $d^{BS}(\mathbf{p}, \mathbf{y}) = (\mathbf{p} - \mathbf{y})^2$.

Let us first calculate the score divergence using method (1). First, we calculate $s(\mathbf{p}, \mathbf{y})$ as follows:

$$\begin{aligned}
s(\mathbf{p}, \mathbf{y}) &= \sum_{k=1}^{K} \phi^{BS}(\mathbf{p}, \mathbf{e}_k) y_k \\
&= \left[(0.4-1)^2 + (0.3-0)^2 + (0.2-0)^2 + (0.1-0)^2\right] \cdot 1 \\
&\quad + \left[(0.4-0)^2 + (0.3-1)^2 + (0.2-0)^2 + (0.1-0)^2\right] \cdot 0 \\
&\quad + \left[(0.4-0)^2 + (0.3-0)^2 + (0.2-1)^2 + (0.1-0)^2\right] \cdot 0 \\
&\quad + \left[(0.4-0)^2 + (0.3-0)^2 + (0.2-0)^2 + (0.1-1)^2\right] \cdot 0 \\
&= 0.5 + 0 + 0 + 0 \\
&= 0.5.
\end{aligned}$$

Let us now calculate $s(\mathbf{y}, \mathbf{y})$ as follows:

$$\begin{aligned}
s(\mathbf{y}, \mathbf{y}) &= \sum_{k=1}^{K} \phi^{BS}(\mathbf{y}, \mathbf{e}_k) y_k \\
&= \left[(1-1)^2 + (0-0)^2 + (0-0)^2 + (0-0)^2\right] \cdot 1 \\
&\quad + \left[(1-0)^2 + (0-1)^2 + (0-0)^2 + (0-0)^2\right] \cdot 0 \\
&\quad + \left[(1-0)^2 + (0-0)^2 + (0-1)^2 + (0-0)^2\right] \cdot 0 \\
&\quad + \left[(1-0)^2 + (0-0)^2 + (0-0)^2 + (0-1)^2\right] \cdot 0 \\
&= 0 + 0 + 0 + 0 \\
&= 0.
\end{aligned}$$

Therefore, using method (1), we derive that:

$$d(\mathbf{p}, \mathbf{y}) = s(\mathbf{p}, \mathbf{y}) - s(\mathbf{y}, \mathbf{y}) = 0.5 - 0 = 0.5.$$

Using method (2), we calculate $d^{BS}(\mathbf{p}, \mathbf{y})$ as follows:

$$\begin{aligned}
d^{BS}(\mathbf{p}, \mathbf{y}) &= \sum_{k=1}^{K} (p_k - y_k)^2 \\
&= (0.4-1)^2 + (0.3-0)^2 + (0.2-0)^2 + (0.1-0)^2 \\
&= 0.36 + 0.09 + 0.04 + 0.01 \\
&= 0.5.
\end{aligned}$$

In this case, the largest value the divergence can take is 2. Whether the outcome of 0.5 is sufficient depends on the application.

4.3 Loss function decomposition

This section further dives into discovering what is optimized in the training procedure of a classifier. Specifically, this section introduces a decomposition of proper scoring rules. This decomposition shows that any proper scoring rule is the sum of three terms, each with an entirely different meaning. After introducing this decomposition, we provide an intuitive explanation of each term. Subsequently, we provide a way to estimate two of these terms that are relevant for finding possible causes of miscalibration.

4.3.1 Calibration-refinement decomposition

Loss functions give the loss of a classifier on a single sample. In practice, however, we are interested in the performance of the classifier on a set of samples. As discussed, the loss on a set of samples is generally calculated as the average sample-wise loss. We refer to this loss as the *empirical loss*. If the average is not taken over a subset of the solution space, such as the training or test set, but over the entire (infinite) solution space, we refer to the loss as the *expected loss*.

Under any proper scoring rule ϕ, the expected loss L of confidence score vector $\mathbf{Z}$ with respect to ground truth vector $\mathbf{Y}$ is given by $L^\phi = \mathbb{E}[\phi(\mathbf{Z}, \mathbf{Y})]$. Theorem 4.3.1 indicates that the expected loss can be decomposed into three terms, namely a calibration, refinement, and dataset loss. This decomposition is often referred to as the calibration-refinement decomposition, and it is based on the work of Kull and Flach [22]. We extend their decomposition to hold for any proper scoring rules instead of only proper scoring rules for which $\phi(\mathbf{Y}, \mathbf{Y}) = 0$. This extension introduces an additional term, namely the dataset loss.

Theorem 4.3.1 (Calibration-refinement decomposition, proven in Appendix B.1)
Let $L^\phi = \mathbb{E}[\phi(\mathbf{Z}, \mathbf{Y})]$ denote the expected loss of confidence score vector $\mathbf{Z}$ with respect to ground truth vector $\mathbf{Y}$ under any proper score rule ϕ. The expected loss L^ϕ can be decomposed as follows:

$$L^\phi = CL^\phi + RL^\phi + DL^\phi, \text{ where } \begin{cases} CL^\phi = \mathbb{E}[d^\phi(\mathbf{Z}, \mathbf{C})] & \text{(Calibration loss)} \\ RL^\phi = \mathbb{E}[d^\phi(\mathbf{C}, \mathbf{Y})] & \text{(Refinement loss)} \\ DL^\phi = \mathbb{E}[\phi(\mathbf{Y}, \mathbf{Y})] & \text{(Dataset loss)} \end{cases}$$

Importantly, the expected loss L^{ϕ}, including the terms in its decomposition, are specific to a particular proper scoring rule ϕ. Therefore, we superscript not only L but also CL, RL, DL, and d with their respective proper scoring rule. For instance, $CL^{BS} = \mathbb{E}[d^{BS}(\mathbf{Z}, \mathbf{C})]$ denotes the calibration loss for the Brier score, whereas $RL^{BS} = \mathbb{E}[d^{BS}(\mathbf{C}, \mathbf{Y})]$ denotes the refinement loss for the Brier score. To improve readability, we omit the superscript whenever the associated proper scoring rule is clear from the context. This decomposition provides valuable and useful insights once one understands each loss term. To that end, we now provide intuition behind each term.

Calibration loss The *calibration loss* is given by $CL = \mathbb{E}[d(\mathbf{Z}, \mathbf{C})]$. Since a score divergence quantifies the difference between two probability distributions, the CL can be viewed as the difference between the confidence score vector $\mathbf{Z}$ and its calibrated score vector $\mathbf{C}$. Therefore, it is natural to interpret the calibration loss CL as a measure of the degree of miscalibration. Since the CL compares the entire vectors, it measures the degree of miscalibration based on multi-class calibration. With this knowledge, the calibration-refinement decomposition informs us that classifiers implicitly optimize (multi-class) calibration during training. This finding suggests that classifiers should improve with regard to calibration throughout the training procedure.

Dataset loss The *dataset loss* is given by $DL = \mathbb{E}[\phi(\mathbf{Y}, \mathbf{Y})]$. For a proper scoring rule ϕ, this loss only depends on the dataset associated with the classification task. Corollary 4.3.1.1 highlights that the dataset loss for both the Brier score and log-loss equals zero. Since these loss functions are the only ones considered throughout this work, the dataset loss is generally ignored. Specifically, we often view the calibration-refinement decomposition as $L = CL + RL$ throughout the remainder of this work.

Corollary 4.3.1.1 (Zero dataset loss, proven in Appendix B.2)
For any proper scoring rule ϕ, the calibration-refinement decomposition is given by $L^{\phi} = CL^{\phi} + RL^{\phi} + DL^{\phi}$ (Theorem 4.3.1). For the Brier score ϕ^{BS} and log-loss ϕ^{LL}, we find that $DL^{BS} = 0$ and $DL^{LL} = 0$, respectively. Therefore, their decomposition can be written as $L^{BS} = CL^{BS} + RL^{BS}$ and $L^{LL} = CL^{LL} + RL^{LL}$, respectively.

Refinement loss The *refinement loss* is given by $RL = \mathbb{E}[d(\mathbf{C}, \mathbf{Y})]$. This loss captures the difference between the calibrated score vector $\mathbf{C} = \mathbb{E}[\mathbf{Y}|\mathbf{Z}]$ and the ground truth vector $\mathbf{Y}$. In contrast to the calibration and dataset loss,

this loss is more challenging to understand intuitively. In order to generate intuition behind the RL, we investigate when it takes either a high or low value. Specifically, we focus on when the refinement loss for a particular confidence score vector takes either a high or low value. To this end, we decompose the refinement loss RL into refinement loss for each unique confidence score vector $\mathbf{z} \in \Delta^K$ by partitioning the solution space of confidence score vectors. Then, the refinement loss can be written as[2]:

$$RL = \sum_{\mathbf{z} \in \Delta^K} P(\mathbf{Z} = \mathbf{z}) \cdot \mathbb{E}\Big[d\Big(\mathbb{E}[\mathbf{Y}|\mathbf{Z} = \mathbf{z}], \mathbf{Y}\Big)\Big].$$

This rewritten formulation shows that the RL is a summation of weighted divergences specific to each unique confidence score vector. In this summation, each unique confidence score vector $\mathbf{z}$ is associated with an expected divergence $\mathbb{E}[d(\mathbb{E}[\mathbf{Y}|\mathbf{Z} = \mathbf{z}], \mathbf{Y})]$. This term can be viewed as the unweighted refinement loss for that particular $\mathbf{z}$. This refinement loss is then weighted by its probability of occurring $P(\mathbf{Z} = \mathbf{z})$.

With this formulation, we can also determine the refinement loss for a particular confidence score vector. In particular, consider a particular confidence score vector $\mathbf{z} \in \Delta^K$. For this $\mathbf{z}$, we denote its weighted refinement loss by:

$$RL_{\mathbf{z}} = P(\mathbf{Z} = \mathbf{z}) \cdot \mathbb{E}\Big[d\Big(\mathbb{E}[\mathbf{Y}|\mathbf{Z} = \mathbf{z}], \mathbf{Y}\Big)\Big].$$

To recap, we would like to obtain intuition behind the refinement loss by investigating the behavior of the refinement loss for a specific confidence score vector. Specifically, we would like to know when the refinement loss for a specific confidence score vector $\mathbf{z}$, denoted by $RL_{\mathbf{z}}$, takes either low or high values. Since $P(\mathbf{Z} = \mathbf{z})$ is only a scaling factor, we disregard this term in our reasoning. The remaining term of importance in $RL_{\mathbf{z}}$ is the expected divergence $\mathbb{E}[d(\mathbb{E}[\mathbf{Y}|\mathbf{Z} = \mathbf{z}], \mathbf{Y})]$. This divergence is minimized if all samples with confidence score vector $\mathbf{z}$ have the same ground truth class. More generally, the divergence is low if (almost) all samples with confidence score vector $\mathbf{z}$ have the same ground truth class. In contrast, the divergence is high if there is much variability among the ground truth class of all samples with confidence score vector $\mathbf{z}$. While the reason behind a low divergence is clear, it might be less clear why high variability in

[2]We write the decomposition of the refinement loss as a sum over confidence score vectors as confidence scores within this vector have finite precision in reality, which leads to finitely many unique confidence score vectors. We could also rewrite this decomposition without assuming finite precision; however, this would substantially degrade the readability. This assumption also plays a role in Equation 4.6.

the ground truth class leads to a high divergence. To that end, consider the following example.

Example 4.3.1 (High variability leads to high refinement loss)

Consider a set of samples with the same confidence score vector that leads to some calibrated score vector. We would like to reason about the divergences between the calibrated score vector and the ground truth vectors of different samples with the same confidence score vector. Let c be the dominant ground truth class in the set of samples. On the one hand, the higher its dominance, the more the calibrated score vector moves towards the ground truth vector associated with class c. For instance, consider a calibrated score of 0.5 for class c. If the number of samples with ground truth class c increases, then the calibrated score of 0.5 increases as well, which leads to an overall lower divergence. On the other hand, the lower its dominance, the more the calibrated score vector moves away from the ground truth vector associated with class c, which leads to a higher divergence. Hence, this example shows that high variability in the ground truth classes of the samples leads to a higher overall divergence than low variability in the ground truth classes.

Hence, for a particular confidence score vector $\mathbf{z}$, $RL_{\mathbf{z}}$ measures how well a classifier can separate samples with different ground truth classes. Put differently, one might interpret the $RL_{\mathbf{z}}$ as measuring the purity of the ground truth classes associated with all samples with the same confidence score vector. Ideally, this confidence score vector is only associated with a single ground truth class. The more samples from different ground truth classes have the same confidence score vector $\mathbf{z}$, the higher the $RL_{\mathbf{z}}$ becomes. Generalizing these findings, the refinement loss RL measures how well a classifier can separate samples of different ground truth classes into the solution space. In the ideal scenario, a confidence score vector is only associated with a single ground truth class. Consequently, the refinement loss penalizes classifiers with low discriminative power.

Since the refinement loss is substantially more challenging to understand than the other two terms, we also provide an example calculation of the refinement loss for a specific confidence score vector in Example 4.3.2. We also illustrate its behavior when varying the calibrated score vector in Example 4.3.3. This latter example is particularly helpful for understanding its behavior.

Example 4.3.2 (Example of the refinement loss)

Consider the setting of a binary classification task with a focus on the refinement loss calculated by the Brier score as $RL^{BS} = \mathbb{E}[d^{BS}(\mathbf{C}, \mathbf{Y})]$.

Suppose we focus on a unique confidence score vector $\mathbf{z}$ that captures ten samples. Out of these samples, one has ground truth class 1 and all other nine have ground truth class 2. In this case, the (empirical) confidence score vector is given by $\mathbf{C} = \mathbb{E}[\mathbf{Y}|\mathbf{Z} = \mathbf{z}] = (1/10, 9/10) = (0.1, 0.9)$. Now, we are interested in the RL^{BS} for this particular $\mathbf{z}$. In order to calculate this quantity, we first decompose the weighted refinement loss $RL^{BS}_{\mathbf{z}}$ into class-wise terms as follows:

$$\begin{aligned}
RL^{BS}_{\mathbf{z}} &= P(\mathbf{Z} = \mathbf{z}) \cdot \mathbb{E}\Big[d^{BS}\big(\mathbb{E}[\mathbf{Y}|\mathbf{Z} = \mathbf{z}], \mathbf{Y}\big)\Big] \\
&= P(\mathbf{Z} = \mathbf{z}) \cdot \sum_{k=1}^{K} \underbrace{P(Y_k|\mathbf{Z} = \mathbf{z})}_{C_k} \cdot \mathbb{E}\Big[d^{BS}\big(\mathbb{E}[\mathbf{Y}|\mathbf{Z} = \mathbf{z}], \mathbf{e}_k\big)\Big] \\
&= P(\mathbf{Z} = \mathbf{z}) \cdot \sum_{k=1}^{K} C_k \cdot \mathbb{E}\Big[d^{BS}\big(\mathbb{E}[\mathbf{Y}|\mathbf{Z} = \mathbf{z}], \mathbf{e}_k\big)\Big],
\end{aligned}$$

where $\mathbf{e}_k$ denotes a vector of length K with a 1 at position k and 0 everywhere else. For convenience, suppose that $P(\mathbf{Z} = \mathbf{z}) = 1$ for our particular $\mathbf{z}$ described above. By plugging in calibrated score vector $\mathbf{C} = (0.1, 0.9)$, we can calculate the (empirical) $RL^{BS}_{\mathbf{z}}$ as follows:

$$\begin{aligned}
RL^{BS}_{\mathbf{z}} &= P(\mathbf{Z} = \mathbf{z}) \cdot \sum_{k=1}^{K} C_k \cdot d^{BS}\big(\mathbb{E}[\mathbf{Y}|\mathbf{Z} = \mathbf{z}], \mathbf{e}_k\big) \\
&= 1 \cdot \sum_{k=1}^{2} C_k \cdot d^{BS}\big(\mathbb{E}[\mathbf{Y}|\mathbf{Z} = \mathbf{z}], \mathbf{e}_k\big) \\
&= \Big(C_1 \cdot d^{BS}\big(\mathbb{E}[\mathbf{Y}|\mathbf{Z} = \mathbf{z}], \mathbf{e}_1\big)\Big) + \Big(C_2 \cdot d^{BS}\big(\mathbb{E}[\mathbf{Y}|\mathbf{Z} = \mathbf{z}], \mathbf{e}_2\big)\Big) \\
&= \Big(0.1 \cdot \big((0.1 - 1)^2 + (0.9 - 0)^2\big)\Big) + \Big(0.9 \cdot \big((0.1 - 0)^2 + (0.9 - 1)^2\big)\Big) \\
&= (0.1 \cdot 1.62) + (0.9 \cdot 0.02) \\
&= 0.162 + 0.018 \\
&= 0.18.
\end{aligned}$$

Hence, the weighted refinement loss for our particular confidence score vector $\mathbf{z}$ is given by $RL^{BS}_{\mathbf{z}} = 0.18$.

Example 4.3.3 (Intuition behind the refinement loss)

In this example, we are interested in better understanding the refinement loss and its behavior for different calibrated score vectors. Consider the setting of Example 4.3.2. Suppose we do not only focus on the confidence score vector $\mathbf{z}$ that leads to $\mathbf{C} = (0.1, 0.9)$ but also on ten other confidence score vectors that lead to ten different calibrated score vectors. In particular, consider eleven confidence score vectors given by $\mathbf{C}_1 = (0.0, 1.0), \mathbf{C}_2 = (0.1, 0.9), \mathbf{C}_3 = (0.2, 0.8), \ldots, \mathbf{C}_{11} = (1.0, 0.0)$. We can now calculate the weighted refinement loss using the Brier score for each calibrated score vector using the procedure described in Example 4.3.2. The following figure illustrates the refinement loss for these vectors as a function of the calibrated score of the first class.

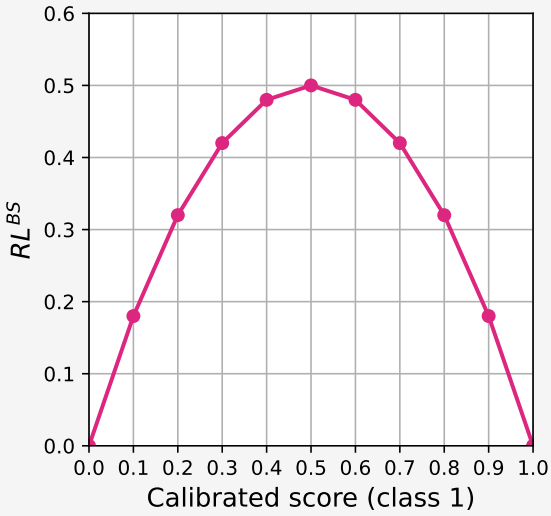


Figure 4.2: RL^{BS} as a function of the calibrated score of class 1.

On the one hand, this figure indicates that the refinement loss is highest when the calibrated score of the first class is 0.5. Since we consider a binary classification problem, this means that the corresponding calibrated score vector is uniformly distributed (*i.e.*, both classes have a confidence score of 0.5). This observation corresponds with our reasoning above that argued that the refinement loss is maximized when the classifier is unable to separate samples with different ground truth classes. On the other hand, this figure also indicates that the refinement loss is lowest when the calibrated score

of the first class is either 0 or 1. This observation also corresponds with our reasoning above that argued that the refinement loss is minimized when the classifiers is perfectly able to separate samples with different ground truth classes.

4.3.2 Estimating the calibration loss

In order to find possible causes of miscalibration, we would like to further inspect the calibration loss. Unfortunately, this loss is much more challenging to calculate than the overall loss on a dataset. This section introduces an estimator for the calibration loss. For simplicity and demonstrative purposes, we focus on the Brier score while proposing the estimator for the calibration loss.

Necessity of binning

The calibration loss for the Brier score is given by $CL^{BS} = \mathbb{E}[d^{BS}(\mathbf{Z}, \mathbf{C})]$ with $\mathbf{C} = \mathbb{E}[\mathbf{Y}|\mathbf{Z}]$. As discussed in § 3.1.1 for confidence calibration, calculating the calibrated score of a specific top-1 confidence score introduces some complications, such as the requirement of binning and estimation. These complications also appear when focusing on the calibration loss. For completeness, let us reintroduce these problems for the calibration loss. In a practical setting, we typically only have a few samples for each unique confidence score vector $\mathbf{Z}$. The reason for this is that each confidence score Z_k in the confidence score vector $\mathbf{Z} = (Z_1, Z_2, \ldots, Z_K)$ can generally take many different values between 0 and 1. (The exact number of unique values depends on the precision of the numbers used throughout the computation.) Based on a set with finitely many samples, it is quite certain that many, if not most, unique confidence score vectors only cover a few samples. Estimating $\mathbf{C} = \mathbb{E}[\mathbf{Y}|\mathbf{Z}]$ for a particular $\mathbf{Z}$ that only covers a few samples is not guaranteed to be accurate. Instead, the calibrated score vector estimated from only a few samples is likely to deviate substantially from the calibrated score vector estimated from infinitely many samples that cover the entire solution space. In order to estimate $\mathbf{C}$ more accurately, there is a necessity to group samples from distinct confidence score vectors by binning on $\mathbf{Z}$. However, binning on K-dimensional vector $\mathbf{Z}$ is generally not that straightforward. Consider the following remark for details.

Remark 4.3.1 (Binning on a K-dimensional vector)

Consider a binning schema in a 1-dimensional setting that splits the entire domain into 5 bins. If we were to extend this binning schema to a 2-dimensional setting, we would already consider $5^2 = 25$ bins. Further-

more, extending this schema to a 3-dimensional setting leads to $5^3 = 125$ bins. The number of bins increases exponentially as the number of dimensions increases. Therefore, binning on a K-dimensional vector tends to lead to a prohibitively high number of bins, assuming the number of bins for each dimension is not exceedingly low. Consequently, a high number of dimensions also leads to a higher number of samples required to make accurate estimations. Furthermore, these samples must be spread roughly evenly across all bins. However, many bins are likely to be empty or only contain very few samples in a realistic setting, which leads to inaccurate estimations.

The remark above indicates that binning on $\mathbf{Z}$ is likely to lead to inaccurate estimations, especially when there are many classes and only a small number of samples. Preferably, we would bin on a 1-dimensional vector, such as the confidence score of a particular class Z_k. Such a binning procedure is possible by assuming that each calibrated score $C_k = \mathbb{E}[Y_k|\mathbf{Z}]$ only depends on the confidence score of class k and not on the confidence scores of other classes. The following relation formalizes this assumption:

$$C_k = \mathbb{E}\big[Y_k|\mathbf{Z}\big] = \mathbb{E}\big[Y_k|Z_k\big]. \tag{4.5}$$

Example 4.3.4 provides an example of this assumption. The idea behind this example is to obtain intuition as to why the assumption seems reasonable.

Example 4.3.4 (Intuition behind Equation 4.5)

Consider a classifier for images of dogs, cats, and horses. Furthermore, assume that the classifier predicts the "dog" class with a confidence score of 0.8 for a given input. For the assumption in Equation 4.5 to hold, the calibrated score for the "dog" class should not depend on how the remaining $1 - 0.8 = 0.2$ confidence score in the prediction is distributed over the other two classes.

We note that, for a perfectly calibrated classifier, the assumption holds automatically.

Remark 4.3.2 (Perfectly calibrated classifiers)

Consider a perfectly calibrated classifier. Under perfect calibration, confidence score vector $\mathbf{Z}$ equals its calibrated score vector $\mathbf{C}$. Accordingly, each confidence score Z_k equals its calibrated score $C_k = \mathbb{E}[Y_k|\mathbf{Z}]$. Because Z_k and C_k are equal, C_k only depends on entry Z_k in the prediction given by $\mathbf{Z}$ (instead of all entries in $\mathbf{Z}$). Since C_k only depends on Z_k, it holds that

$C_k = \mathbb{E}[Y_k|Z_k]$. Consequently, $\mathbb{E}[Y_k|Z_k] = \mathbb{E}[Y_k|\mathbf{Z}]$. Therefore, the assumption formalized by Equation 4.5 holds for a perfectly calibrated classifier. Notably, the objective while training a classifier with a proper scoring rule is to obtain a perfectly calibrated classifier with perfect accuracy. Hence, the assumption generally holds more and more throughout training as calibration is optimized.

This assumption is also related to the relaxation of multi-class calibration to class-wise calibration. Notably, most literature focuses on class-wise calibration instead of multi-class calibration when requiring the entire confidence score vector $\mathbf{Z}$ to be well-calibrated [11, 23, 37, 40, 55]. As argued, the definition of multi-class calibration is relaxed to class-wise calibration with assumption $C_k = \mathbb{E}[Y_k|Z_k]$ (instead of $\mathbb{E}[Y_k|\mathbf{Z}]$). Hence, a direct connection exists between the assumption in Equation 4.5 and the evaluation of miscalibration found in the literature.

Due to the previously mentioned reason, we assume Equation 4.5 to hold. To that end, we use $C_k = \mathbb{E}[Y_k|Z_k]$ in the remainder of this section. Let us return to estimating the CL using the Brier score. In order to estimate the CL, we first further decompose it into class-wise terms as follows:

$$\begin{aligned}
CL &= \mathbb{E}[d^{BS}(\mathbf{Z}, \mathbf{C})] && \text{(Theorem 4.3.1)} \\
&= \mathbb{E}\left[\sum_{k=1}^{K}(Z_k - C_k)^2\right] \\
&= \sum_{k=1}^{K}\mathbb{E}\Big[(Z_k - C_k)^2\Big] \\
&= \sum_{k=1}^{K}\underbrace{\mathbb{E}\Big[\big(Z_k - \mathbb{E}[Y_k|Z_k]\big)^2\Big]}_{CL_k} && (C_k = \mathbb{E}[Y_k|Z_k])
\end{aligned}$$

where CL_k represents the class-wise calibration loss for class k. Using the definition of an expectation, the class-wise calibration loss CL_k can be written as a weighted sum over all unique confidence scores $p \in [0, 1]$:

$$\begin{aligned}
CL_k &= \mathbb{E}\Big[\big(Z_k - \mathbb{E}[Y_k|Z_k]\big)^2\Big] \\
&= \sum_{p\in[0,1]} P\big(Z_k = p\big) \cdot \big(p - \mathbb{E}[Y_k|Z_k = p]\big)^2. \qquad (4.6)
\end{aligned}$$

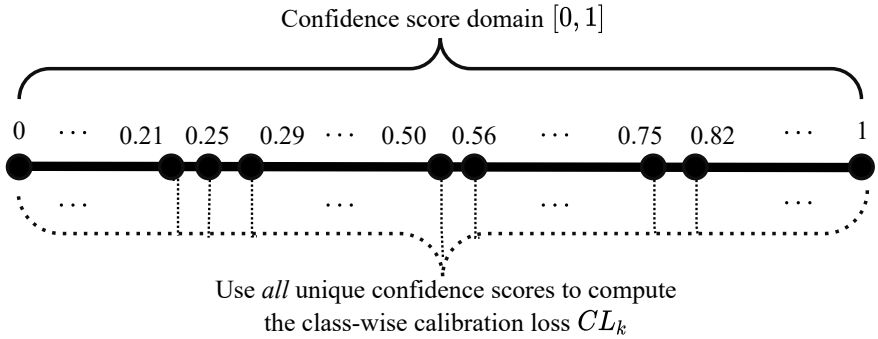


Figure 4.3: An illustration of the confidence score domain $[0, 1]$ with nine example confidence scores.

To exemplify this formula, consider Figure 4.3. This figure illustrates the confidence domain $[0, 1]$ together with nine particular instances of unique confidence scores, considering a precision of two decimals. Naturally, the space of unique confidence scores is much larger. In this figure, each unique confidence score can be associated with many different samples. For instance, the confidence score of 0.25 might cover several different samples. We would now like to estimate the class-wise calibration loss for class k. Therefore, we need to estimate the calibrated score $\mathbb{E}[Y_k|Z_k]$ for each unique confidence score $Z_k \in [0, 1]$ (by abuse of notation, we use Z_k instead of $Z_k = p$ for some unique confidence score p). Similar to the situation with $\mathbf{Z}$, each unique confidence score Z_k is likely to only capture very few samples, albeit more than $\mathbf{Z}$. As discussed previously, in order to estimate a representative calibrated score, we can apply a binning procedure that groups samples. Since the confidence score Z_k is 1-dimensional, we can apply such a procedure without obtaining a high number of empty bins. To that end, we partition the confidence score domain $[0, 1]$ into M disjoint bins:

$$\mathcal{B}_k = \{B_{k,1}, B_{k,2}, \ldots, B_{k,M}\}, \tag{4.7}$$

where each bin $B_{k,m}$ contains all samples in the $[b_{m-1}, b_m)$ interval. Then, bin $B_{k,1}$ captures all samples in the $[b_1 = 0, b_2)$ interval and bin $B_{k,M}$ captures all samples in the $[b_{M-1}, b_M = 1]$ interval. Since the intervals of the bins do not depend on the class, we assume equivalent intervals are used for each class. An example of a binning procedure is illustrated in Figure 4.4. In this figure, the first bin denotes $B_{k,1} = [0.0, 0.1)$, the second bin denotes $B_{k,2} = [0.1, 0.2)$, and so forth. Note that this figure only shows the bins for some example confidence

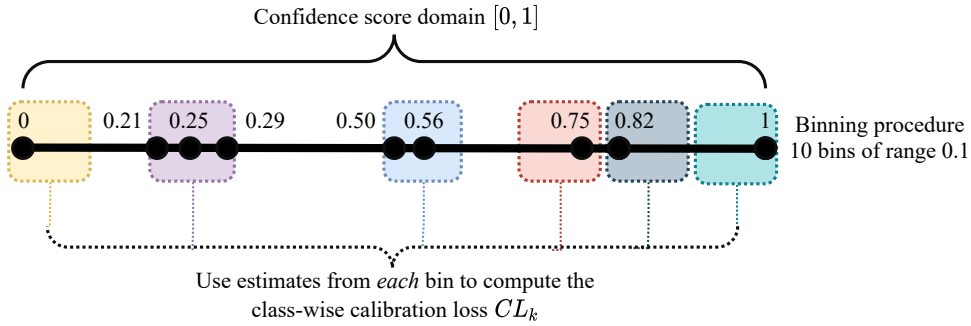


Figure 4.4: An illustration of a possible binning procedure on the confidence score domain $[0, 1]$ with nine example confidence scores. This binning procedure creates

scores. After applying such a binning procedure, we can start estimating the class-wise calibration loss CL_k based on a set of samples.

Finitely many samples

As argued previously, we have finitely many samples in a practical setting. Therefore, expectations have to be estimated by calculating averages. Suppose we have a dataset consisting of N samples whose input-label pairs are denoted by $\{x^{(i)}, \mathbf{y}^{(i)}\}_{i=1}^{N}$. These samples lead to N confidence score vectors $\{\mathbf{z}^{(i)}\}_{i=1}^{N}$, one for each input-label pair. In order to estimate Equation 4.6 using the binning procedure described above, we have to estimate the average confidence score and calibrated score in each bin. In this case, each bin $B_{k,m}$ contains all samples whose confidence score for the kth class (*i.e.*, z_k) falls into the $[b_{m-1}, b_m)$ interval. Then, we can estimate the confidence and calibrated score in each bin by computing averages across all samples in each bin. For a particular class k, we compute the average confidence score for bin m as $\bar{z}_{k,m} = \sum_{j \in B_{k,m}} z_k^{(j)} / |B_{k,m}|$. Similarly, we compute the average calibrated score for bin m as $\bar{c}_{k,m} = \sum_{j \in B_{k,m}} y_k^{(j)} / |B_{k,m}|$. Using these averages, the estimator for the CL_k can then be written as a sum over all M bins:

$$\widehat{CL}_k(\mathcal{B}_k) = \sum_{m=1}^{M} \frac{|B_{k,m}|}{N} (\bar{z}_{k,m} - \bar{c}_{k,m})^2, \tag{4.8}$$

where $|B_{k,m}|$ denotes the number of samples in bin $B_{k,m}$ and $|B_{k,m}|/N$ is a weighing factor that is decided based on the proportion of samples in bin $B_{k,m}$.

In order to compute the estimated calibration loss $\widehat{CL}$, the class-wise estimator $\widehat{CL}_k$ for each class k can be summed. Given some set of binning schemes $\mathcal{B} = \{\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_K\}$ (see Equation 4.7 for an example of $\mathcal{B}_k$), the estimated calibration loss is given by summing the class-wise calibration loss estimate for each class:

$$\widehat{CL}(\mathcal{B}) = \sum_{k=1}^{K} \widehat{CL}_k(\mathcal{B}_k). \tag{4.9}$$

Throughout the remainder of this work, any value given for the calibration loss CL is estimated with $\widehat{CL}(\mathcal{B})$. An estimator for the RL can be derived in a similar fashion. The primary differences between these estimators are the terms inside the score divergence. Appendix B.3 provides an example of the estimator for the RL. Similar to the conf-ECE$_2$ and CW-ECE$_2$, we use 15 equally-ranged bins to compute the CL and RL. In this section, we derived an estimator for the calibration loss under the Brier score. An estimator for the calibration loss under other proper scoring rules can be derived in a similar fashion. The only difference is a change in the score divergence d.

Estimation errors

Unfortunately, the estimator proposed for the CL is not guaranteed to compute an accurate calibration loss due to estimation errors. Both the binning procedure and estimation from finite samples can introduce estimation errors. The severity of the estimation errors seems to depend on the loss function used to compute the CL. Example 4.3.5 provides a comparison of the Brier score and log-loss concerning sensitivity to estimation errors.

Example 4.3.5 (Estimation errors in loss functions)

Consider a binary classification task where some classifier predicts class 1 with a confidence score of 0.5. This prediction is given by $\mathbf{Z} = (0.5, 0.5)$. In order to investigate possible estimation errors, we investigate the behavior of divergence $d(\mathbf{Z}, \mathbf{C})$ for different instances of $\mathbf{C}$. We focus on the individual contribution of class 1 to $d(\mathbf{Z}, \mathbf{C})$. Figure 4.5 plots the contribution of class 1 to $d(\mathbf{Z}, \mathbf{C})$ as a function of different calibrated scores for class 1. This contribution to the divergence is shown for both the Brier score and log-loss. Since the confidence score of class 1 is 0.5, the divergence component of this class should be 0 at a calibrated score of 0.5. For a calibrated score of 0.5, we observe that the divergence is, indeed, 0 for both the Brier score and log-loss.

Furthermore, we observe that the log-loss has a relatively high gradient at this score, whereas the Brier score has a gradient of 0. Consequently, small estimation errors might have a substantial impact on the calibration loss estimated for the log-loss. In contrast, small estimation errors might be negligible while estimating the calibration loss under the Brier score as this gradient is 0. Therefore, the Brier score seems much less sensitive to estimation errors on the calibrated score vector than the log-loss.

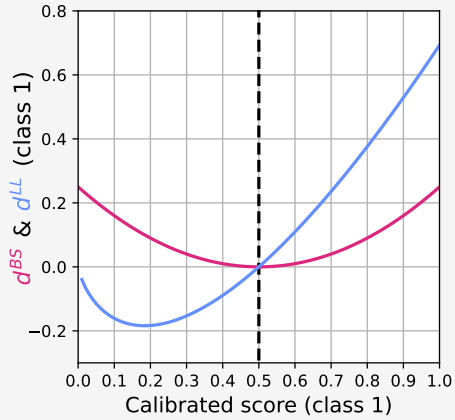
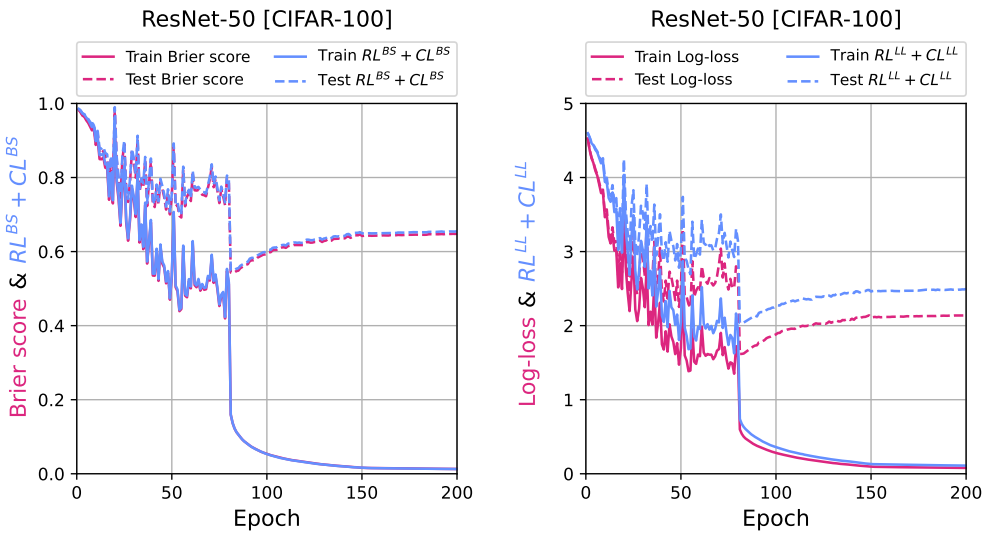


Figure 4.5: Comparison of the divergence component between a predicted confidence score of 0.5 and calibrated scores, ranging from 0 to 1 for class 1, shown for both the Brier score (d^{BS}) and log-loss (d^{LL}).

The example above suggests that the Brier score might be substantially less sensitive to estimation errors than the log-loss while estimating the calibration loss. This belief is further supported by Figure 4.6. This figure plots the Brier score against the estimated Brier score (*i.e.*, $CL^{BS} + RL^{BS}$), and the log-loss against the estimated log-loss (*i.e.*, $CL^{LL} + RL^{LL}$) while training a classifier. Details on the training procedure are given in Appendix A.3. In this figure, the Brier score and estimated Brier score seem to align perfectly for all epochs. In contrast, there seems to be a difference between the log-loss and the estimated log-loss. However, we notice that the log-loss and estimated log-loss show a high positive correlation, especially after epoch 80. Their major difference is some seemingly constant factor. The seemingly constant difference for the log-loss would be an interesting future direction to investigate. More comparisons of the loss function and its estimated variant based on the calibration and refinement loss are provided in Appendix C.3. Since the Brier score seems substantially less sensitive to errors than the log-loss for the estimation of the calibration loss, we continue with this loss function for the remainder of this chapter.



(a) **(b)**

Figure 4.6: Comparison of the Brier score against the estimated Brier score (left), and the log-loss against the estimated log-loss (right) for ResNet-50 on CIFAR-100.

4.4 Classification objectives and loss function terms

In the first section of this chapter, we argued that a classification task consists of a correctness and calibration objective. Afterward, we decomposed the loss function used to train a classifier into three terms, namely a calibration, refinement, and dataset loss. On the one hand, we are now aware of what the loss function minimizes throughout the training procedure of a classifier. On the other hand, we also know what we would like to have optimized, namely the correctness and calibration objective. Based on this knowledge, we move to the next phase in Figure 4.1 and relate the refinement and calibration loss to the correctness and calibration objective. Specifically, we couple the correctness objective to the refinement loss and the calibration objective to the calibration loss. The relation between these objectives and losses provides us with a more profound understanding regarding the optimization of the objectives during training. This understanding will ultimately lead to two hypotheses for the origination of miscalibration in the next section.

4.4.1 Correctness objective and refinement loss

The first compelling relation appears between the correctness objective and refinement loss. In § 4.1.1, we stated that the correctness objective requires minimization of the expected classification error $\mathbb{E}[\mathbb{1}(\arg\max_k Z_k \neq \arg\max_k Y_k)]$. In contrast, the training procedure generally minimizes the refinement loss $RL = \mathbb{E}[d(\mathbf{C}, \mathbf{Y})]$. Although the classification error and refinement loss might seem different, a strong relationship exists under a minor assumption. Let us provide intuition behind this relation with an illustrative example.

Example 4.4.1 (Classification error and refinement loss)

Consider the setting proposed in Example 4.3.2. That is, a binary classification task with a focus on the refinement loss calculated by the Brier score as $RL^{BS} = \mathbb{E}[d^{BS}(\mathbf{C}, \mathbf{Y})]$. To illustrate the relation between the classification error and RL^{BS}, we would like to plot the values of these quantities for different calibrated score vectors.

Similar to Example 4.3.2, we consider eleven calibrated score vectors $\mathbf{C}_1 = (0.0, 1.0), \mathbf{C}_2 = (0.1, 0.9), \ldots, \mathbf{C}_{11} = (1.0, 0.0)$. We can then compute the RL^{BS} for each calibrated score vector by following the procedure in Example 4.3.2. In this example, we can derive the classification error from the calibrated score vector under a minor assumption. For each unique $\mathbf{Z}$ that lead to the calibrated scores $\mathbf{C}_1, \mathbf{C}_2, \ldots, \mathbf{C}_{11}$, let us assume that the index of the top-1 confidence score in $\mathbf{Z}$ corresponds with the index of the top-1 calibrated score in its corresponding $\mathbf{C}$ (*i.e.*, $\arg\max_k C_k = \arg\max_k Z_k$). In words, this assumption requires that the predicted class of a particular confidence score vector equals the most common class in the calibrated score vector. (We later highlight that this assumption is reasonable.) Under this assumption, the accuracy on the set of samples with confidence score $\mathbf{Z}$ is given as $\max_k C_k$. Consequently, the classification error is given as $1 - \max_k C_k$. For instance, the accuracy of $\mathbf{C}_2 = (0.1, 0.9)$ is 0.9 and its respective classification error is 0.1.

The figure below illustrates the RL^{BS} and classification error for $\mathbf{C}_1 = (0.0, 1.0), \mathbf{C}_2 = (0.1, 0.9), \ldots, \mathbf{C}_{11} = (1.0, 0.0)$ as a function of the calibrated score of the first class.

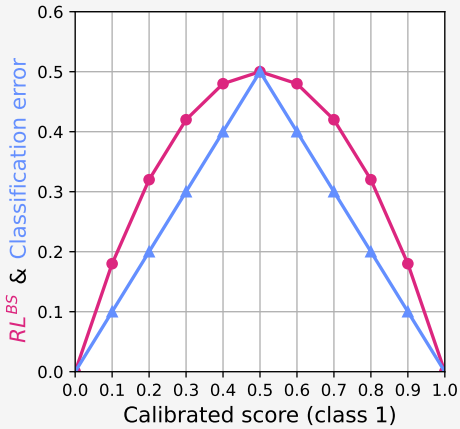


Figure 4.7: RL^{BS} and classification error as a function of the calibrated score of class 1.

This figure highlights that the RL^{BS} and classification error exhibit similar behavior. For instance, high values for the RL^{BS} correspond to high classification errors. Similarly, low values for the RL^{BS} correspond to low classification errors. Hence, the RL^{BS} and classification error seem to behave similarly.

The example above indicates a strong relationship between the correctness objective and refinement loss under a minor assumption. For a confidence and calibrated score vector, this assumption requires the class of the top-1 confidence score to correspond to the class of the top-1 calibrated score. Consequently, this assumption is violated when the top-1 class in the confidence score vector deviates from the top-1 class in the calibrated score vector. Under perfect confidence calibration, the top-1 confidence score aligns with the top-1 calibrated score. Therefore, their classes will be equivalent as well. Even if the top-1 confidence score is not perfectly calibrated under confidence calibration, the assumption still holds as long as the top-1 classes are equivalent. In other words, the assumption is only violated when the confidence calibration is (very) poor. Since the calibration loss is minimized throughout training, we believe the calibration to improve, including the confidence calibration that strengthens the assumption. Under this reasonable assumption, there seems to be a strong connection between the correctness objective and refinement loss.

Next to the illustrative example above, we provide empirical evidence implying a strong relation between the correctness objective and refinement loss. Specifically, we plot the classification error and RL^{BS} throughout the train-

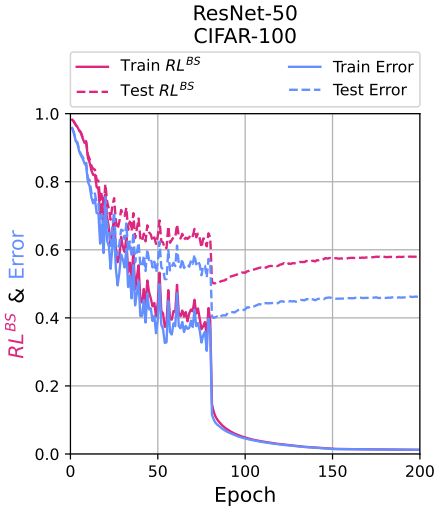


Figure 4.8: Comparison of the classification error and RL for a ResNet-50 model on CIFAR-100.

ing procedure on both the training and test set for a particular classifier and dataset. This comparison is illustrated in Figure 4.8. This figure shows a high positive correlation between the classification error and refinement loss as their patterns are nearly identical. Appendix C.2 provides additional results for several datasets and neural network classifiers. In all cases, we find a high positive correlation between the classification error and classification error. To conclude, both the theoretical and empirical views indicate that the correctness objective and refinement loss are tightly connected.

4.4.2 Calibration objective and calibration loss

The second interesting relation exists between the calibration objective and calibration loss. In § 4.1.1, we stated that the calibration objective is to minimize $\text{MC-CE}_2 = (\mathbb{E}[|\mathbf{Z} - \mathbf{C}|^2])^{\frac{1}{2}} = (\mathbb{E}[\sum_{k=1}^{K}(Z_k - C_k)^2])^{\frac{1}{2}}$ (see Equation 4.1). In contrast, the training procedure generally minimizes the calibration loss $CL^{\phi} = \mathbb{E}[d^{\phi}(\mathbf{Z}, \mathbf{C})]$ under proper scoring rule ϕ. The only difference between these two quantities is how they penalize differences between $\mathbf{Z}$ and $\mathbf{C}$. For instance, consider the calibration loss under the Brier score that is given by $CL^{BS} = \mathbb{E}[d^{BS}(\mathbf{Z}, \mathbf{C})] = \mathbb{E}[\sum_{k=1}^{K}(Z_k - C_k)^2]$. In this case, the only difference between these two quantities is a square root. In other words, we find that $CL^{BS} = \text{MC-CE}_2^2$.

For completeness, we also empirically show that the behavior of CL^{BS} is closely connected with the behavior of popular calibration metrics. As indicated above, the CL^{BS} is equivalent to the MC-CE$_2^2$. Similar to the conf-ECE$_2$ and CW-ECE$_2$, the MC-CE$_2$ is more sensitive to poorly calibrated confidence scores (see Chapter 3 for details). Therefore, it seems more natural to compare the CL^{BS} with the conf-ECE$_2$ and CW-ECE$_2$ instead of the conf-ECE and CW-ECE. In order to show that the behavior of CL^{BS} is closely connected with the behavior of CW-ECE$_2$ and conf-ECE$_2$, we visualize their behavior throughout training. Intuitively, we expect CL^{BS} and CW-ECE$_2$ to show a stronger correlation than CL^{BS} and conf-ECE$_2$ as the latter term only measures confidence miscalibration.

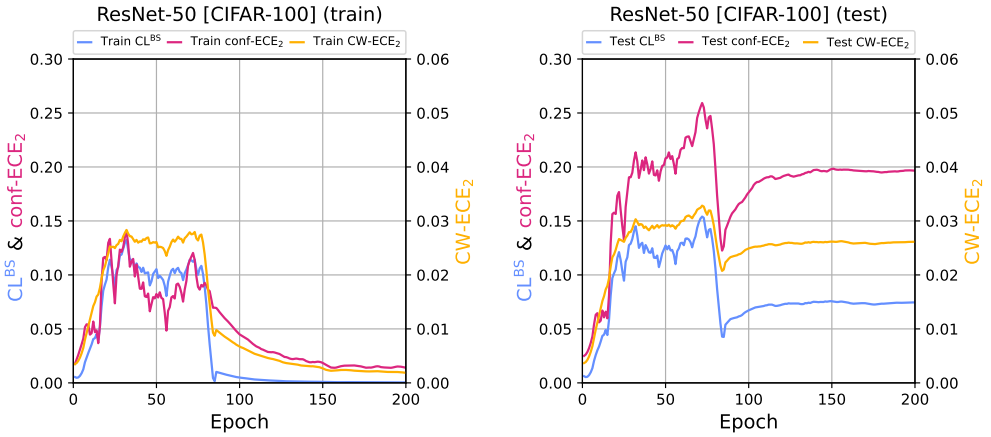
Remark 4.4.1 (Smoothing filter)

Before introducing the diagrams showing the relationship between these calibration metrics, we first remark the following. Initially, the learning rate is relatively high, which results in rather noisy outcomes for the calibration metrics. Since we would like to focus on trends in the diagrams, we apply the Savitzky-Golay Smoothing filter with a window of eleven [44]. This filter smoothes the curves while preserving their trend. Importantly, we use this filter for all diagrams whose x-axis represents the number of epochs unless otherwise specified.

Figure 4.9 illustrates the behavior of the calibration metrics on a specific training and test set. We observe similar overall trends in each diagram with some slight deviations. These deviations are easily explainable as the terms measure a different interpretation of calibration. The conf-ECE$_2$ deviates most from the other calibration metrics as it measures confidence miscalibration instead of either class-wise or multi-class calibration. To conclude, both the theoretical and empirical views indicate that the calibration metrics and calibration loss are tightly connected.

4.5 Hypotheses for the origination of miscalibration

The previous section highlights two interesting findings, namely (1) minimizing the refinement loss is strongly connected to the correctness objective and (2) minimizing the calibration loss is strongly connected to the calibration objective. The second objective indicates that classifiers should improve concerning calibration throughout the training procedure. In § 3.3, however, we observed that numerous classifiers are miscalibrated. In this section, we move towards the next phase in Figure 4.1. That is, we further dive into these seemingly contradicting



(a) **(b)**

Figure 4.9: Comparison of the CL^{BS}, conf-ECE$_2$, and CW-ECE$_2$ for a ResNet-50 model on CIFAR-100 on the (a) training and (b) test set, using the smoothing filter.

findings based on a relatively simple but crucial observation. Based on this observation, we present two hypotheses for the origination of miscalibration.

4.5.1 Errors in models

Although our second finding indicates that calibration is optimized during calibration, we should not overlook that the optimization occurs on a *training set*. In contrast, the evaluation that generally shows miscalibration of classifiers is performed on a *test set*. Therefore, the degree of miscalibration might be minor on the training set but substantial on the test set. Consequently, the source of miscalibration might be rooted in differences between the training and test set. In particular, its source might be errors in models whose causes are (at least partly) rooted in (slight) differences in the training and test set distribution. To that end, we investigate the relation between these errors and miscalibration. Before investigating their relation, we first introduce two fundamental but common errors whose underlying cause is related to (slight) differences in the training and test set. Afterward, we investigate how these issues interact with our findings.

Overfitting

A fundamental issue in machine learning is classifiers not generalizing well from observed data to unobserved data [53]. This issue is typically referred to as *overfitting*. When overfitting, classifiers perform very well on the training set with respect to the loss function but fit poorly on the test set. Typically, classifiers increasingly improve on the training set during the training. There is typically a tipping point in the training process on the test set, after which the performance will deteriorate. This phenomenon tends to occur when a classifier trains for too long or has more trainable parameters than can be justified by the data [7]. In those cases, the classifier can start learning irrelevant patterns or spurious relations within the training dataset and use those as a basis for predictions. Consequently, although the classifier might perform exceptionally well on the training data, it fails to generalize well to the test set. We refer to [53] for a more complete and in-depth overview regarding possible causes of overfitting.

Ultimately, overfitting is a fundamental issue in machine learning that prohibits classifiers from generalizing well to unseen data. In order to reduce the effect of overfitting, multiple techniques have been proposed. Nevertheless, overfitting tends to appear to at least some degree for high-capacity classifiers, even when using mitigation techniques.

Distribution shift

Next to overfitting, *distribution shift* is another common issue in machine learning. Briefly, a distribution shift appears when the training and test set distributions differ from each other [41]. For instance, a distribution shift appears when a training set consists of images of animals, whereas the test set contains cartoon images of those animals. In this case, the two sets of images come from different distributions. Although different types of distribution shifts exist, we focus on the general idea of dissimilar distributions in this work. That is, distribution shifts are not limited to strictly different types of images. A distribution shift might also appear when it comes to uncertainty. The uncertainty on the training set could be substantially lower than on the test set. Like overfitting, a distribution shift leads to a classifier that performs relatively well on its training set but notably worse on its test set. Since the classifier is trained on the training set, its performance will improve throughout the training procedure. Under a distribution shift, the test set distribution substantially differs from the training set distribution. Therefore, the trained classifier will most likely perform substantially worse on the test set.

4.5.2 Hypotheses for the origination of miscalibration

We hypothesized that the causes of miscalibration might be rooted in differences between the training and test set. To that end, we introduced two fundamental issues in machine learning that appear due to (slight) differences between the training and test set. This section concretizes our suspicion of miscalibration stemming from differences between training and test set. Specifically, we propose two hypotheses based on the fundamental issues introduced in the previous section.

Miscalibration due to overfitting

In the previous section, we argued that overfitting is a fundamental issue in machine learning. We generally observe overfitting by inspecting the values taken by the loss function on the training and test set. For the expected loss L under any proper scoring rule ϕ, we proved that $L = RL + CL + DS$. For the Brier score and log-loss, we observed that $DS = 0$, which means that $L = RL + CL$. Furthermore, we argued that the RL and CL are directly connected to the correctness and calibration objective, respectively. Additionally, we formalized both of these objectives as classification problems. Therefore, we can interpret RL and CL as cost functions for these classification problems. Let us consider a classifier that starts overfitting to ϕ on the training set. In this case, we know that L will be relatively low on the training set but substantially higher on the test set. Since the RL and CL can be interpreted as cost functions for classification problems, we expect overfitting to be noticeable for both the RL and CL. In other words, we expect the RL and CL to be low on the training set but significantly higher on the test set. Consequently, we expect the overfitting to be noticeable for both the classification error and calibration metrics. Based on this reason, we derive our first hypothesis for the origination of miscalibration: *"We hypothesize that miscalibration on the test set (at least partly) stems from overfitting to the loss function on the training set"*. Throughout the remainder of this work, we refer to this hypothesis as the *overfitting hypothesis*.

Miscalibration due to a distribution shift

As highlighted by the calibration objective, the predicted confidence score vector should equal its calibrated score vector. To that end, predictions of classifiers should not be overconfident or underconfident. In order for a classifier to learn how confident it should be, they must train on a training set that is representative of the uncertainty present in the test set (*i.e.*, on unseen data). However, the training set is typically not representative of this uncertainty. In particular,

a classifier generally has significantly higher accuracy on the training set than on the test set. Consequently, a classifier generally learns to be relatively confident during training, which is appropriate for that comparatively high accuracy. However, this learned confidence is generally too high for the test set, as the accuracy on the test set is much lower. Therefore, predictions on the test set will tend to be overconfident on average. Ultimately, we can interpret this phenomenon as a distribution shift concerning uncertainty between the training and test set. The uncertainty on the training set is not representative of the uncertainty on the test set. After training, there is much less uncertainty on the training set than on the test set.

This suspected phenomenon primarily appears when a significant gap exists between the accuracy computed on the training and test set. As argued, overfitting is a major cause of this behavior. Therefore, we suspect this phenomenon to appear when a classifier overfits to its loss function. While the overfitting hypothesis would already degrade calibration, we expect this phenomenon to degrade calibration even further. Coupling this to the loss function decomposition, we expect that a substantial gap in the RL found between the training and test set further worsens the CL on the test set. We view this as a negative feedback loop. From this reasoning, we derive the following hypothesis: *"We hypothesize that miscalibration is further degraded when a classifier overfits to the loss function due to a distribution shift in the uncertainty between the training and test set"*. Following this hypothesis, we expect the degree of miscalibration to be impacted more substantially than the classification error for classifiers that overfit during training. Throughout the remainder of this work, we refer to this hypothesis as the *distribution shift hypothesis*. Unfortunately, this hypothesis alone would be challenging to verify because it appears as a consequence of overfitting to the loss function. Therefore, we need to assess both hypotheses jointly.

4.6 Experimental evaluation

In the previous section, we proposed two hypotheses for the origination of miscalibration. In this section, we verify these hypotheses empirically with various experiments. Preferably, we would like to test each hypothesis in isolation. However, designing such experiments seems impossible as the distribution shift hypothesis occurs due to the overfitting hypothesis. Hence, we have to investigate the combination of both hypotheses.

4.6.1 Statistics of interest

Let us first focus on the overfitting hypothesis. To recap, this hypothesis states that miscalibration on the test set stems (at least partly) from overfitting to the loss function on the training set. In order to test this hypothesis, we study the training process of a classifier that suffers from overfitting. To what extent a training process is overfitting can be measured from the gap between the loss on the training set and test set. If our overfitting hypothesis is correct, we would expect an increase in miscalibration when the training starts to overfit. Due to the calibration-refinement decomposition, we focus on the calibration loss under the Brier score as the calibration metric to test this hypothesis.

To summarize, we would like to investigate plots of the loss function and CL^{BS} on the training and test set throughout the training procedure in order to verify the overfitting hypothesis. In order to test the distribution shift hypothesis, we would like to test whether the CL^{BS} is impacted more than the classification error when the classifier starts overfitting to the loss function. Therefore, we extend the quantities of interest for testing the first hypothesis with the classification error. Consequently, we would now like to investigate plots of the classification error, loss function, and CL^{BS} on the training and test set throughout the training procedure.

4.6.2 Datasets

Although the quantities of interest are known, relevant datasets for the experiments have yet to be chosen. For our experiments, we focus on several datasets to ensure the stability of our results across different problem domains and configurations. In particular, we focus on CIFAR-10 [20], CIFAR-100 [20], and ImageWoof[3]. In addition to popular benchmark datasets CIFAR-10 and CIFAR-100, we included ImageWoof because we believe it is more representative of a real-world use case. ImageWoof is a subset of ten classes from ImageNet [6] that are all dog breeds, which makes it relatively challenging to differentiate between classes. Furthermore, the images in ImageWoof are substantially larger than the images in both CIFAR-10 and CIFAR-100. Details regarding these datasets, including the image sizes, number of classes, and number of samples for each set, can be found in Appendix A.1.

[3]The ImageWoof dataset can be found at https://github.com/fastai/imagenette.

4.6.3 Architecture selection

Although the quantities of interest and relevant datasets are known, model architectures have yet to be chosen. We decided to repeat our experiment on different instances of the popular ResNet architecture [12] with increasing complexity to more easily compare how model complexity relates to calibration. We consider ResNet-14, ResNet-32, ResNet-50, ResNet-110, and ResNet-152 throughout experiments. Details regarding the training procedure can be found in Appendix A.3. Notably, we train all models for 200 epochs using the Brier score. As argued earlier, we focus on measuring miscalibration with the CL^{BS} using 15 equally-ranged bins. Therefore, we opt for the Brier score as the loss function. Initially, all models start with a learning rate of 0.1, which is decreased by a factor of 10 at epoch 80. The learning rate is further decreased by a factor of 10 at epoch 150. This learning rate scheduler aligns with the one proposed by the original ResNet architecture [12]. Notably, we will perform our experiments without data augmentation as we are primarily interested in the behavior of these models while overfitting.

4.6.4 Results

With the selected datasets and architectures, we now test the hypotheses by plotting the classification error, Brier score, and CL^{BS} over the epochs. For conciseness and clarity, we focus on a single dataset, namely CIFAR-100. Additional experiments for CIFAR-10 and ImageWoof are included in Appendix C.4. The additional experiments in the appendix show similar findings as this section. Figure 4.10 illustrates the diagrams mentioned in the previous sections for ResNet-14, ResNet-32, ResNet-50, and ResNet-110 on CIFAR-100. We exclude ResNet-152 from this figure for conciseness as this architecture showed roughly identical patterns as ResNet-110.

4.6.5 Discussion

Based on Figure 4.10, we now discuss whether the results are in line with the hypotheses we formulated. We focus mainly on the behavior of the curves after epoch 80, which is when the learning rate is dropped by a factor of ten from 0.1 to 0.01. This focus stems from the curves showing much volatility before this epoch, which is unsuitable for proper comparison. After around epoch 80, we observe that all classifiers start overfitting to the loss function (*i.e.*, the Brier score). In particular, the Brier score on the training set improves, whereas it degrades on the test set. Observing overfitting is essential as it is a prerequisite for the hypotheses. For all classifiers, we observe that overfitting to the loss

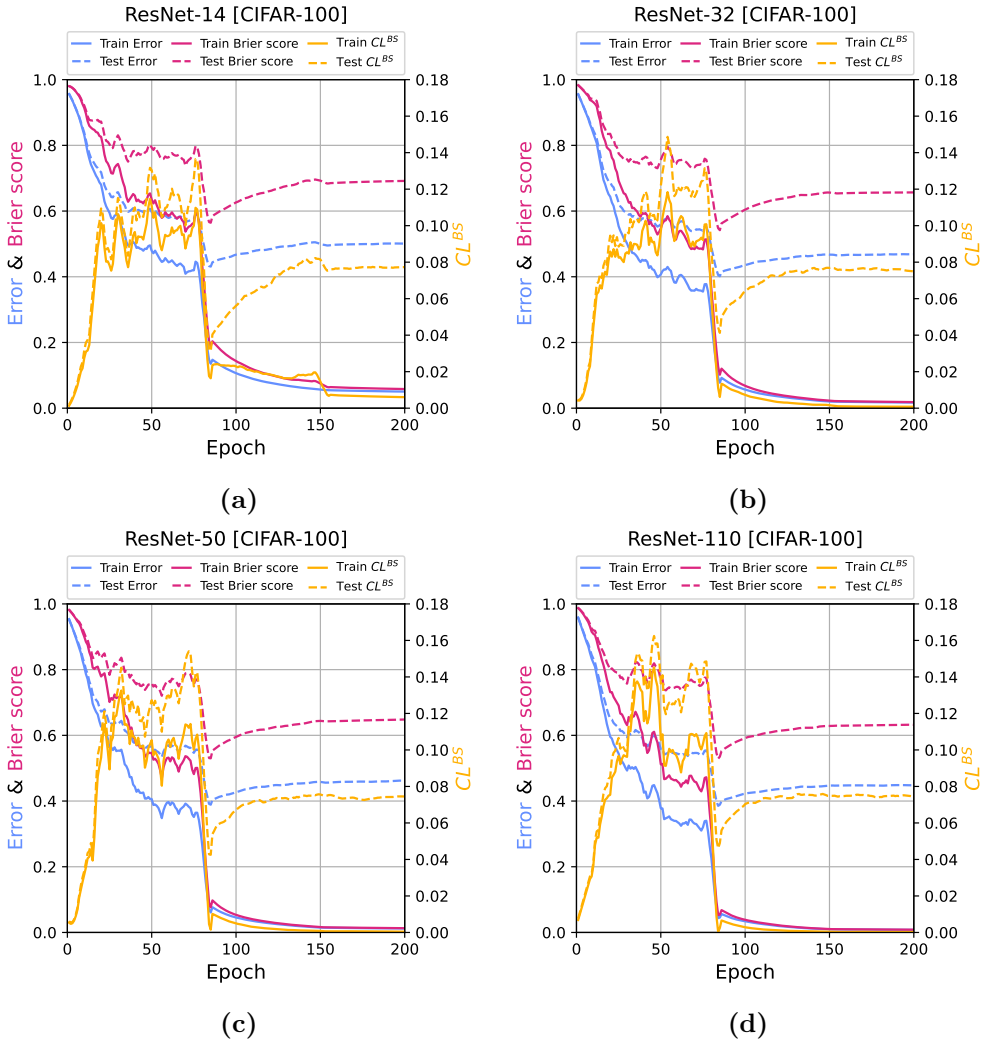


Figure 4.10: Comparison of the classification error, Brier score, and CL^{BS} for (a) ResNet-14, (b) ResNet-32, (c) Resnet-50, and (d) ResNet-110 on both the training and test set for CIFAR-100, using the filter.

function goes paired with a degrading CL^{BS} on the test set. As the classifiers overfit to the loss function, the CL^{BS} on the training set improves, whereas the CL^{BS} on the test set degrades. Since we observe this behavior for all classifiers, our overfitting hypothesis seems to be correct. This claim is strengthened by the additional results proposed in Appendix C.4.

We now shift our focus to the distribution shift hypothesis. For this hypothesis, we would like to check whether the CL^{BS} is impacted more substantially than the classification error when the classifier starts overfitting to the loss function. We observe this specific behavior for all classifiers after around epoch 80. While the classification error on the test set degrades, the CL^{BS} degrades relatively more. Let us focus on the behavior of ResNet-50 on the test set to exemplify this claim. Here, the classification error equals 0.4028 at epoch 81, whereas it becomes 0.4567 at epoch 150. In contrast, the CL^{BS} equals 0.0442 at epoch 81, whereas it becomes 0.0754 at epoch 150. For the classification error, this is an increase of roughly 13.38%, whereas the increase for the CL^{BS} is roughly 70.73%. In other words, the CL^{BS} becomes substantially worse compared to the classification error. We observe similar behavior for all other classifiers. The relative degradation in CL^{BS} is always much more substantial than the degradation in classification error while the classifier overfits. Therefore, we find that the CL^{BS} is impacted more substantially than the classification error when the classifier starts overfitting to the loss function. This observation supports the distribution shift hypothesis. This claim is further strengthened by additional results proposed in Appendix C.4. Despite being unable to estimate the calibration loss under the log-loss accurately, we also investigate the hypotheses for classifiers trained with the log-loss in Appendix C.5. Since we are unable to estimate CL^{LL} accurately, we instead compare the classification error and log-loss against the conf-ECE_2 and CW-ECE_2, as these metrics also measure the degree of miscalibration. This appendix also finds support for the overfitting and distribution shift hypotheses. Hence, we also find support for these hypotheses for classifiers trained with the log-loss instead of the Brier score.

Therefore, the experiments build evidence that overfitting and its related distribution shift are important causes for the miscalibration of classifiers. Using this knowledge, we may now reason about possible remedies against miscalibration.

4.7 Notes on mislabeled samples

To conclude this chapter, we would like to present a subtle but important remark concerning calibration that, to the best of our knowledge, has not yet been mentioned in the literature on calibration. The calibration metrics listed in this work measure the degree of miscalibration by quantifying the deviation between the confidence score vector $\mathbf{Z}$ and calibrated score vector $\mathbf{C} = \mathbb{E}[\mathbf{Y}|\mathbf{Z}]$. We find that a subtle but important assumption rests in the ground truth vector $\mathbf{Y}$ of the samples with confidence score vector $\mathbf{Z}$. Specifically, we assume that

these samples are labeled correctly. In other words, we assume that the ground truth vector of each sample indicates the actual ground truth label of that sample. However, this is not always the case due to samples often being labeled manually by humans [36]. Some samples might be mislabeled as humans are prone to making mistakes. For instance, a human might label an image of a dog as a wolf or even as a cat. Northcutt et al. [36] show that numerous popular benchmark datasets contain at least some degree of mislabeled samples. For instance, 5.83% of all samples in the validation set of the popular ImageNet dataset are mislabeled.

Mislabeled samples can be problematic for evaluating the calibration of classifiers. Let us elaborate on this statement with an example. Assume that a set of samples with a particular confidence score vector $\mathbf{Z} = (0.8, 0.2)$ is correctly labeled, *i.e.*, the ground truth vector of each sample indicates the actual class of that sample. For perfect calibration of $\mathbf{Z} = (0.8, 0.2)$, we would like 80% of all samples with this prediction to be of class 1 (*i.e.*, have ground truth vector $\mathbf{Y} = (1, 0)$) and 20% of all samples with this prediction to be of class 2 (*i.e.*, have ground truth vector $\mathbf{Y} = (0, 1)$). Then, its calibrated score vector is given by $\mathbf{C} = (0.8, 0.2)$. In this case, the prediction is perfectly calibrated under all notions of calibration as $\mathbf{Z} = \mathbf{C}$. Additionally, the classifier achieves an accuracy of 80% on these samples. However, suppose all samples that were thought of as being of class 2 have been mislabeled. In reality, these samples are actually of class 1 instead of class 2. Then, the classifier is perfectly calibrated for $\mathbf{Z}$, despite 20% of all samples being mislabeled. If the samples had been labeled correctly, the classifier would have achieved an accuracy of 100% and its calibrated score vector would have been $\mathbf{C} = (1, 0)$. Then, the classier that is now perfectly calibrated with the mislabeled samples would not be perfectly calibrated if the mislabeled samples were corrected. This observation might lead to calling a classifier perfectly calibrated despite it not actually being perfectly calibrated due to mislabeled samples.

Let us return to the setting where the calibrated score vector for $\mathbf{Z} = (0.8, 0.2)$ is given by $\mathbf{C} = (0.8, 0.2)$, even though 20% of all samples associated with this prediction are incorrectly labeled as class 2 instead of class 1. Suppose that the classifier would have predicted $\mathbf{Z} = (1, 0)$ instead of $\mathbf{Z} = (0.8, 0.2)$ for all these samples. Then, this prediction seems miscalibrated as $\mathbf{Z} = (1, 0)$ deviates from $\mathbf{C} = (0.8, 0.2)$. However, if the samples that were identified as class 2 would have been labeled correctly, then the calibrated score vector would have been given by $\mathbf{C} = (1, 0)$. In this case, the classifier would have been perfectly calibrated for this prediction. Hence, a classifier might seem miscalibrated due to mislabeled samples while being perfectly calibrated with correctly labeled samples.

Therefore, perfect calibration means that the confidence score vector $\mathbf{Z}$ matches the calibrated score vector $\mathbf{C} = \mathbb{E}[\mathbf{Y}|\mathbf{Z}]$, where the ground truth vector $\mathbf{Y}$ of the samples corresponds with how the labeling was performed (often manually by humans) and not necessarily with the actual ground truth labels. Perfect calibration being defined as such can be undesirable in safety-critical applications. We do, however, believe that assuming the ground truth vectors correspond with the actual ground truth labels is reasonable. We believe this to be reasonable as the labeling task is always performed with the intention of correctly labeling the samples. Furthermore, labeling is often crowd-sourced [36]. This technique seems likely to minimize the number of mislabeled samples as it is less sensitive to errors than letting a single person label all samples. Additionally, we believe that the labeling of samples receives more attention in safety-critical applications as correct labels are more important in these applications. Consequently, we would think this leads to fewer (or none) mislabeled samples in the applications where calibration is important. In addition, we would like to note that the discussed impact of mislabeled training data may not necessarily be a bad thing. An example is given in Remark 4.7.1. Surprisingly, we were unable to find any discussion on this topic in the literature surrounding calibration. Therefore, it would be interesting to further investigate how mislabeled samples impact the notion of calibration on multiple classifiers and datasets with a varying degree of mislabeled samples.

Remark 4.7.1 (Reflecting the labeler's uncertainty)

Suppose the existence of some labeling procedure where a doctor labels X-ray images as either "no pneumonia", "viral pneumonia", or "bacterial pneumonia". In this situation, a doctor might be uncertain about the actual class of an image. However, the doctor generally still has to label the image as one of the three classes. In this case, the uncertainty associated with the labeling procedure is not directly visible in the assigned class label.

Suppose that the doctor labels 80 images as "viral pneumonia" and 20 images as "no pneumonia". Furthermore, suppose that these 20 images labeled as "no pneumonia" are incorrectly labeled and actually belong to the "viral pneumonia" class. Additionally, suppose that some classifier is trained on a training set that includes these images. We assume that this classifier is perfectly multi-class calibrated after training and predicts the same confidence score vector of (0.2, 0.8, 0.0) for all these 100 images. (The classifier does not predict this confidence score vector for any other image.) Since the classifier is perfectly multi-class calibrated, the calibrated score vector of the previously mentioned confidence score vector is (0.2, 0.8, 0.0)

as well. In this example, the classifier achieves an accuracy of 80% on the 100 samples. If there were no mislabeled images, the calibrated score vector would equal (0.0, 1.0, 0.0) instead for this scenario (and the accuracy would increase from 80% to 100%). In a sense, the calibrated score vector in the situation with mislabeled images captures the doctor's uncertainty on those 20 images it labeled incorrectly. Put differently, the classifier tries to mimic the labeling of the doctor as well as possible. By doing so, a perfectly calibrated classifier somewhat reflects the uncertainty of the labeling procedure that manifests from mislabeled images. In other words, the deviation between the calibrated score vector with mislabeled samples and without mislabeled samples somewhat indicates the labeler's uncertainty in the labeling procedure. Intuitively, this does not necessarily seem like a bad thing. Note that this only holds for mislabeled images where the labeler has a challenging time selecting a class and not for mislabeled images that stem from a labeler mislabeling an image on purpose.

Chapter 5

Prevention of miscalibration

In the last chapter, we found that classifiers tend to be miscalibrated. Through numerous experiments, we found support for the hypothesis that overfitting is an important cause of miscalibration. Based on this finding, a natural next step is to reason about the remedies for miscalibration.

Remedies for miscalibration can either be mitigative or preventative. On the one hand, preventative techniques are applied *while training* a classifier. These techniques might prevent high degrees of miscalibration. As hinted in the previous chapter, limiting the degree of overfitting seems promising for preventing high degrees of miscalibration. In this chapter, we focus on various preventative techniques with an emphasis on techniques that limit overfitting. Note that many more techniques exist than captured in this chapter.

On the other hand, mitigative techniques are applied *after training* a classifier. These ad-hoc techniques generally aim to align the confidence score vector with its corresponding calibrated score vector. These types of techniques will be the focus of Chapter 6.

5.1 Data augmentation

The previous chapter hinted that overfitting to the loss function is a cause of miscalibration. Hence, the degree of miscalibration may be reduced by ensuring classifiers overfit less to the loss function. Consequently, any technique for preventing or reducing overfitting is expected to reduce the degree of miscalibration. Note, however, that overfitting is a challenging problem, particularly for complex classifiers [1]. This means that we can expect at most some reduction in miscalibration. Whether this is sufficient depends on the use case.

A commonly employed technique to reduce overfitting is *data augmentation* [45]. This umbrella term captures any concrete technique used to (artificially) increase the amount of data by producing (slightly) modified copies of existing copies [30]. Since this technique tends to reduce the degree of overfitting, it also seems a promising candidate for reducing the degree of miscalibration. In

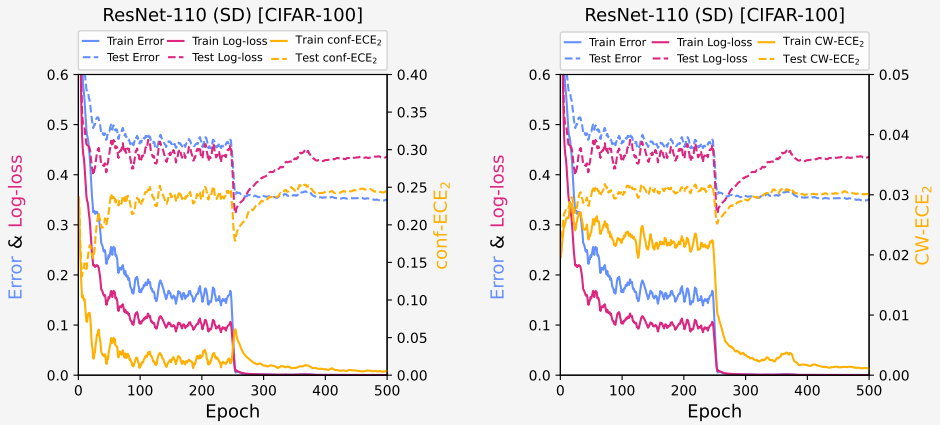
our setting, possible modifications of images can be produced by geometric transformations, color modifications, and mixing multiple images.

5.2 Classifier selection strategy

An entirely different technique that might limit miscalibration is rooted in changing the performance criterion used to select a classifier after training. As highlighted in the previous chapter, classification tasks consist of a correctness and calibration objective. Practitioners generally select the classifier with the highest accuracy on a validation set. By doing so, they strictly focus on the correctness objective in the classification task. The calibration of this classifier can be substantially worse than a classifier chosen based on a different criterion. Hence, one might want to select a performance criterion that captures both classification objectives. In this case, the loss computed by the loss function would be a natural choice since this loss can be decomposed into a calibration and refinement loss. Therefore, we suspect that selecting the loss as the performance criterion can lead to significantly better-calibrated classifiers. To illustrate this remark, consider the following example.

Example 5.2.1 (Loss function as performance criterion)

We trained a ResNet-110 with Stochastic Depth on CIFAR-100 using the log-loss for 500 epochs with a drop in learning rate from 0.1 to 0.01 at epoch 250 and a drop from 0.01 to 0.001 at epoch 350. Furthermore, we used light data augmentation throughout training (see Appendix A.4 for details). Figure 5.1 illustrates the training curves of this classifier. In a practical setting, a practitioner would select a classifier from the training procedure with the highest accuracy on the validation set. In this case, the highest classification accuracy on the validation set was reached at epoch 497. In contrast, suppose the loss is used as the performance criterion instead. Then, the classifier associated with the lowest loss on the validation set would be chosen. This classifier can be found at epoch 252.



(a) **(b)**

Figure 5.1: Training curves of the Resnet-110 (SD) on CIFAR-100 using the log-loss for 500 epochs. Figure 5.1a shows the conf-ECE$_2$ and Figure 5.1b shows the CW-ECE$_2$.

Table 5.1 indicates the performance statistics for the correctness and calibration objective associated with these two epochs. (We show how different conf-ECE$_2$ values translate to different confidence reliability diagrams in § 7.2.1.) The statistics in this table indicate that selecting the loss as the performance criterion leads to an improvement of 28.34% in the conf-ECE$_2$. Furthermore, the improvement for the CW-ECE$_2$ is 17.16%. Therefore, selecting the classifier with the loss as the performance criterion leads to a substantially better-calibrated classifier. However, it does also lead to lower classification accuracy. In particular, the classification accuracy for the classifier chosen based on the loss is 3.35% lower. Whether such a decrease in accuracy is acceptable depends on the particular use case associated with the classification task. However, this observation is captivating as it shows that calibration can be improved (substantially) by selecting the classifier based on the lowest loss instead of the highest accuracy on the validation set.

Table 5.1: Performance statistics for the epoch with the highest accuracy on the validation set (epoch 497) and the epoch with the lowest loss on the validation set (epoch 252). Upwards arrows indicate that higher values are better and downwards arrows indicate that lower values are better.

Epoch	Set	Accuracy (↑)	conf-ECE$_2$ (↓)	CW-ECE$_2$ (↓)
497	Validation	0.6612	0.2361	0.0331
497	Test	0.6512	0.2438	0.0303
252	Validation	0.6254	0.1821	0.0297
252	Test	0.6294	0.1747	0.0251

5.3 Modifying the loss function

Another promising technique consists of modifying the loss function. For instance, a regularization term can be added to the loss function [33]. Regularization terms tend to reduce overfitting. Examples of such regularization terms are L1 and L2 regularization. On the one hand, L1 regularization can oppose overfitting by reducing trainable parameters towards zero. On the other hand, L2 regularization can oppose overfitting by pushing weights to be small but not necessarily zero.

5.4 Label smoothing

Label smoothing [47] is a technique that assigns a small portion of the ground truth class's probability mass to other classes. Many works in the literature have empirically shown that this technique can reduce overfitting [4]. Pereyra et al. [38] argue that label smoothing helps reduce overconfidence confidence scores. Therefore, it seems like a promising candidate for reducing the degree of miscalibration. Müller et al. [32] empirically show that this method can, indeed, lead to better-calibrated classifiers.

Chapter 6

Mitigation of miscalibration

In contrast to preventing miscalibration during training, it is also possible to mitigate miscalibration after training. In this case, the training procedure of the classifiers does not need to be modified, which introduces various benefits. For instance, applying such a technique does not require an existing classifier to be retrained. This benefit is particularly advantageous for classifiers that underwent a lengthy and expensive training procedure.

6.1 Post-hoc calibration methods

Miscalibration can be mitigated after training with so-called *post-hoc calibration methods*. A post-hoc calibration method aims to learn a calibration map that can be applied to the *logits* or non-probabilistic output of the classifier. Throughout this work, we denote the *logits* of a confidence score vector $\mathbf{Z}$ as $\mathbf{Z}^L \in \mathbb{R}^K$. The confidence score vector $\mathbf{Z}$ is derived from its logits $\mathbf{Z}^L$ by applying a Softmax function σ_{SM}, *i.e.*, $\mathbf{Z} = \sigma_{\text{SM}}(\mathbf{Z}^L)$. (The inner workings of the Softmax function become apparent in Equation 6.1.) A calibration map is then applied to $\mathbf{Z}^L$ with the objective of obtaining a confidence score vector that is better calibrated than $\mathbf{Z}$. Formally, such a calibration map is given by $\hat{\pi} : \mathbb{R}^K \to \Delta^K$. A calibration map $\hat{\pi}$ is then applied as $\hat{\pi}(\mathbf{Z}^L)$ on top of the logits $\mathbf{Z}^L$. Generally, we refer to $\hat{\pi}(\mathbf{Z}^L)$ as the re-calibrated confidence score vector of $\mathbf{Z}$. Ideally, the re-calibrated confidence score vector $\hat{\pi}(\mathbf{Z}^L)$ equals its calibrated score vector $\mathbf{C}$, *i.e.*, $\hat{\pi}(\mathbf{Z}^L) = \mathbf{C}$. Therefore, the objective of each post-hoc calibration method is to estimate the *canonical calibration map* π for which $\pi(\mathbf{Z}^L) = \mathbf{C}$. Some post-hoc calibration methods directly take confidence score vector $\mathbf{Z}$ as input instead of the logits $\mathbf{Z}^L$. By slight abuse of notation, we allow the calibration maps of those methods to take the confidence score vector as input.

Each post-hoc calibration method contains one or more parameters that shape its calibration map $\hat{\pi}$. These parameters are generally optimized using either the Brier score or log-loss on a hold-out set. Throughout this work, we refer to this hold-out set as the *calibration set*. Formally, such a set of c image-label pairs

is given by $\mathcal{D}_{\text{cal}} = ((X^{(1)}, Y^{(1)}), (X^{(2)}, Y^{(2)}), \ldots, (X^{(c)}, Y^{(c)})$. After learning the parameters from a hold-out set, the post-hoc calibration map can be applied to confidence score vectors on other samples (*e.g.*, samples from the test set). For clarity, Figure 6.1 illustrates the entire workflow revolving post-hoc calibration methods.

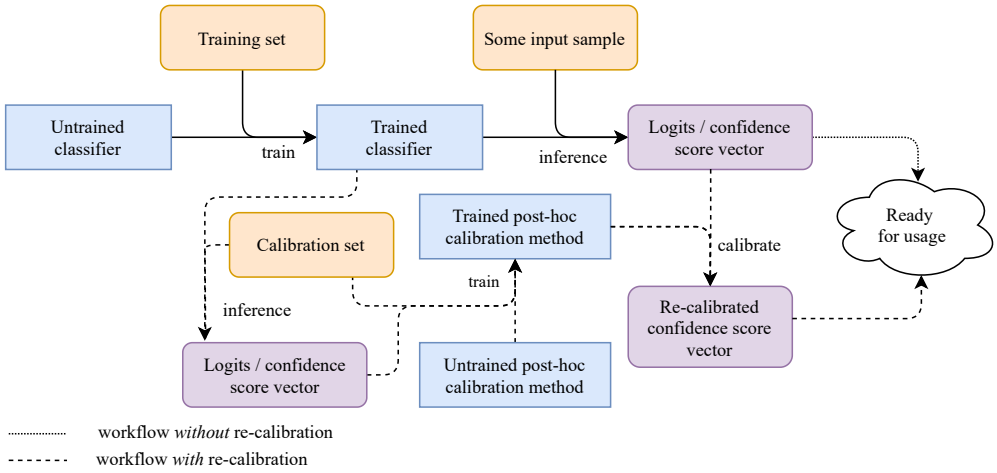


Figure 6.1: Illustration of the post-hoc calibration workflow. For comparison, both the workflow with and without the usage of post-hoc calibration are depicted.

6.2 Challenges of post-hoc calibration methods

Before introducing various post-hoc calibration methods, we first highlight some challenges that these methods face. The first challenge is rooted in possible differences between the calibration and test set. The parameters of post-hoc calibration methods are optimized based on the calibration set. Consequently, the post-hoc calibration method only works well under the assumption that the distribution of the calibration set is (roughly) equivalent to the distribution of the test set (or any set on which inference is performed). Therefore, the size and quality of the calibration set can influence the performance of the post-hoc calibration method.

Even if the calibration and test set distribution are equivalent, the confidence score vectors generated on the calibration set might still deviate from those generated on the test set. For instance, the test set might produce confidence score vectors that were never produced for the calibration set. In order to re-

calibrate these vectors appropriately, the post-hoc calibration method needs to make assumptions about the shape of the calibration map. Possible assumptions exist around the smoothness and monotonicity.

Additionally, we highlighted that the parameters of post-hoc calibration methods are generally optimized using the Brier score or log-loss on the calibration set. If a post-hoc calibration method consists of many samples, it can start overfitting to the loss function on the calibration set. Then, the method can perfectly re-calibrate the confidence score vectors in the calibration set while not being able to generalize well to the test set. Therefore, post-hoc calibration methods generally use very few parameters or make distributional assumptions. In the latter case, we speak of *parametric* post-hoc calibration methods. These methods assume some underlying distribution on the relationship between confidence and calibrated score vectors. For instance, a parametric method might assume a sigmoidal relationship between these two vectors. Admittedly, such a distributional assumption limits the possible complexity of the calibration map. Consequently, these methods might be outperformed by non-parametric methods (*i.e.*, methods that do not make distributional assumptions). Generally, post-hoc calibration can be split into binary and multi-class, depending on the number of classes present in the classification task.

6.3 Binary post-hoc calibration

We first introduce post-hoc calibration in a binary setting. The objective of a binary post-hoc calibration method is to learn a calibration map $\hat{\pi}$ that transforms a confidence score vector $\mathbf{Z}$ into its calibrated score vector $\mathbf{C}$. Let $q = \arg\max_k Z_k$ denote the predicted class. Then, Z_q denotes the confidence score of the predicted class and Z_q^L denotes its logit. For convenience, we assume that $\hat{\pi}$ only takes the logit of the predicted class Z_q^L as input. Then, it outputs the re-calibrated score of the predicted class $\hat{\pi}(Z_q^L)$. The re-calibrated confidence score of the other class can then be computed as $1 - \hat{\pi}(Z_q^L)$. The methods introduced in this section and the subsequent section are primarily derived from the work of Guo et al. [9] and Wenger et al. [50].

Platt scaling Originally, Platt et al. [39] introduced *platt scaling* in the context of *Support Vector Machines* (SVMs) to transform outputs ranging from $[-\infty, +\infty]$ to probabilities in $[0, 1]$. Presently, Platt scaling also makes a parametric calibration method for binary classification tasks. This method fits a

logistic regression classifier to the confidence scores. Formally, Platt scaling computes the re-calibrated confidence score $\hat{\pi}(Z_q^L)$ for logit Z_q^L as follows:

$$\hat{\pi}(Z_q^L) = \frac{1}{1 + \exp\left(-w \cdot Z_q^L - b\right)},$$

where $w \in \mathbb{R}$ represents a slope and $b \in \mathbb{R}$ an intercept. Most commonly, these parameters are learned by minimizing the log-loss on the calibration set. However, one might also opt for another loss function instead. Additionally, one might consider using a random or grid search over the parameter values and select those that minimize some calibration metric. For the post-hoc calibrators in this work, we stick to the most commonly employed method of finding parameters by minimizing the log-loss on the calibration loss.

Platt scaling is efficient as it only needs to learn two parameters during training and is easily computable during inference. However, this method assumes a parametric, sigmoidal relationship between confidence scores and calibrated scores [2]. Unfortunately, such an assumption does not necessarily hold. For instance, applying Platt scaling to already well-calibrated confidence scores results in miscalibrated scores because Platt scaling cannot learn the identity calibration map.

Histogram binning *Histogram binning* [54] is a straightforward non-parametric calibration method. This method divides the confidence scores into a fixed number of mutually exclusive bins $B_1, B_2, \ldots, B_M$ with corresponding bin boundaries $[b_0 = 0, b_1), [b_1, b_2), \ldots [b_{M-1}, b_M = 1]$. Each bin B_m contains all samples whose confidence score falls in range $[b_{m-1}, b_m)$. Then, the average accuracy across all samples in each bin is computed, which gives rise to a set of average accuracies $\{a_m\}_{m=1}^M$. These accuracies represent the re-calibrated confidence scores associated with a particular bin. Formally, histogram binning computes the re-calibrated confidence score $\hat{\pi}(Z_q)$ for confidence score Z_q as follows:

$$\hat{\pi}(Z_q) = \sum_{m=1}^{M} \mathbb{1}(Z_q \in B_m) \cdot a_m,$$

where $\mathbb{1}$ denotes the indicator function (*i.e.*, it is 1 if $Z_q \in B_m$ and 0 otherwise). Notably, the bin boundaries can be determined in various manners. Typically, they are either chosen as equal length intervals (*i.e.*, equal length in the confidence domain $[0, 1]$) or equal size intervals (*i.e.*, equal number of samples in each bin). The two parameters relevant for this method are the number of bins and the binning scheme.

Isotonic regression *Isotonic regression* [9] relaxes the assumption made by Platt scaling. Specifically, it relaxes the assumption of a sigmoidal relationship between confidence scores and calibrated scores to an isotonic (*i.e.*, non-decreasing) relationship. Isotonic regression learns a piece-wise constant function f_{IR} that transforms the confidence scores into re-calibrated confidence scores by minimizing the square loss between $f_{\text{IR}}(Z_q)$ and the ground truth label Y_q on the calibration set.

Beta calibration *Beta calibration* is parametric method for calibration that is designed explicitly for classifiers with output range $[0, 1]$ [21]. This method allows for calibration maps that capture Beta distributions. In contrast to the sigmoidal relationship captured by Platt scaling, Beta calibration allows for a richer family of calibration maps. Formally, Beta calibration computes the re-calibrated confidence score $\hat{\pi}(Z_q)$ for confidence score Z_q as follows:

$$\hat{\pi}(Z_q) = \frac{1}{1 + \exp\left(-w_1 \cdot \ln Z_q - w_2 \cdot \ln\left(1 - Z_q\right) - b\right)},$$

where $w_1, w_2 \in \mathbb{R}$ represent slopes and $b \in \mathbb{R}$ represents an intercept. After learning these parameters by minimizing the log-loss on the calibration set, they are fixed during inference. Although allowing for a richer family of calibration maps, this method is still limited by its distributional assumption.

6.4 Multi-class post-hoc calibration

In this section, we shift the focus from binary to multi-class post-hoc calibration methods. In contrast to binary methods that can only deal with classification tasks with two classes, these methods can deal with a classification task consisting of two or more classes.

Extension of binary calibration methods Until recently, no native multi-class post-hoc calibration methods were popular. All popular methods for multi-class calibration were extensions of binary methods applied in a one-vs-rest fashion. Specifically, Zadrozny and Elkan [55] proposed to decompose a multi-class classification task into K binary classification tasks. Each binary classification task treats a specific class k as the positive class and all other classes combined as the negative class. Then, each binary classification task can be tackled with a single binary classifier. Consequently, the decomposition of K binary classification tasks leads to K binary classifiers, one for each class. One can then apply a binary post-hoc calibration method to each binary classifier.

Focusing on the positive class, we then obtain a single confidence score for each binary classifier. Merging these confidence scores in a single vector, we obtain an unnormalized vector consisting of K re-calibrated confidence scores. After normalizing this vector with the sum of all re-calibrated confidence scores, we obtain the normalized re-calibrated confidence score vector whose elements sum to 1 and fall in the $[0, 1]$ domain. This approach can be used for Platt scaling, histogram binning, isotonic regression, and Beta calibration.

Temperature scaling *Temperature scaling* [9] is a straightforward multi-class extension of Platt scaling. Temperature scaling replaces the two scalar parameters by a single scalar parameter $T > 0$, often called the *temperature*. The temperature parameter is generally found by minimizing the log-loss on the calibration set. With this temperature parameter, temperature scaling computes the re-calibrated confidence score vector $\hat{\pi}(\mathbf{Z}^L)$ for confidence score vector as:

$$\hat{\pi}(\mathbf{Z}^L) = \sigma_{\text{SM}}\left(\frac{\mathbf{Z}^L}{T}\right) = \frac{\exp(\mathbf{Z}^L/T)}{\sum_{k=1}^{K}\exp(\mathbf{Z}_k^L/T)}, \tag{6.1}$$

where $\mathbf{Z}_k^L \in \mathbb{R}$ denotes the logit value for class k. These functions are applied are applied component-wise. To get a deeper understanding on the temperature parameter, consider the following example.

Example 6.4.1 (Intuition behind temperature parameter)
Let us consider how the value of T influences the confidence scores in the re-calibrated confidence score vector. As T approaches 0, the confidence score of the predicted class collapses to a point mass (*i.e.*, it becomes 1). Once $T = 1$, we recover the original confidence scores that would be obtained by applying the Softmax function σ_{SM} to the logits. However, with $T > 1$, the confidence scores are "softened". More specifically, as T approaches ∞, the confidence scores approach $1/K$, representing maximum uncertainty.

Due to the monotonicity of the logistic function σ_{SM}, temperature scaling does not modify the predicted class. Consequently, temperature scaling preserves the accuracy of the original classifier, which is a desirable property for any post-hoc calibration method. However, since temperature scaling only has a single parameter, its family of calibration maps is restricted in their flexibility. Therefore, this method is unable to learn complex calibration maps, which can be a significant limitation.

Matrix scaling Similar to temperature scaling, *matrix scaling* is a multi-class extension of Platt scaling. In contrast to temperature scaling, matrix scaling performs a linear transformation to the logits $\mathbf{Z}^L$ using a matrix $W \in \mathbb{R}^{K \times K}$ and vector $b \in \mathbb{R}^K$. Formally, matrix scaling computes the re-calibrated confidence score vector as:

$$\hat{\pi}(\mathbf{Z}^L) = \sigma_{\text{SM}}\big(W \cdot \mathbf{Z}^L + b\big) = \frac{\exp(W \cdot \mathbf{Z}^L + b)}{\sum_{k=1}^{K} \exp(W \cdot \mathbf{Z}^L + b)_k}.$$

Parameters W and b are learned on the calibration set by minimizing the log-loss. After learning these parameters on the calibration data, they are fixed during inference. Since matrix scaling consists of more parameters than temperature scaling, it has more flexibility in expressing complex relationships between logits and re-calibrated confidence score vectors. However, since it has access to many parameters, it might also overfit on the calibration set.

Vector scaling Similar to temperature scaling and matrix scaling, *vector scaling* is a multi-class extension of Platt scaling. Vector scaling is similar to matrix scaling. The only difference is that vector scaling restricts matrix W to be a diagonal matrix [9]. We denote this diagonal matrix as $W' \in \mathbb{R}^{K \times K}$. Formally, vector scaling computes the re-calibrated confidence score vector as:

$$\hat{\pi}(\mathbf{Z}^L) = \sigma_{\text{SM}}\big(W' \cdot \mathbf{Z}^L + b\big) = \frac{\exp(W' \cdot \mathbf{Z}^L + b)}{\sum_{k=1}^{K} \exp(W' \cdot \mathbf{Z}^L + b)_k},$$

After learning W' (only the diagonal values) and b by minimizing the log-loss on the calibration set, they are fixed during inference. Since vector scaling consists of more parameters than temperature scaling, it has more flexibility in expressing complex relationships between logits and re-calibrated confidence score vectors. Compared to matrix scaling, it has fewer parameters, which might prevent it from overfitting on the calibration set.

Chapter 7

Evaluation of calibration methods

In the previous two chapters, we proposed various countermeasures against miscalibration in the form of preventative and mitigative calibration methods. In this chapter, we study the effectiveness of these methods empirically on a variety of classifiers and datasets. In order to apply the mitigative calibration methods in an easy-to-use manner, we designed and implemented a Python library that contains all post-hoc calibration methods. This library also allows for the evaluation of calibration in a straightforward manner.

7.1 Library for post-hoc calibration and calibration evaluation

In order to experiment with different post-hoc calibration methods and evaluate calibration, we created a Python library called `uncertainty estimation`. This library provides functionality to apply all of the post-hoc calibration methods described in Chapter 6. Additionally, it provides an easy API for evaluating the degree of miscalibration using all calibration metrics described in Chapter 3. This library is designed to be easily extendable regarding not only post-hoc calibration methods but also calibration metrics. Example 7.1.1 provides a brief example of how the isotonic regression post-hoc calibration method can be applied to the output of a classifier, together with the evaluation of the re-calibrated output using the conf-ECE_2.

Example 7.1.1 (Post-hoc calibration and calibration evaluation)

Suppose we posses a training, validation, and test set consisting of input images, which are denoted by `x_train`, `x_val`, and `x_test`, respectively. The ground truth labels of these sets are given by `y_train`, `y_val`, and `y_test`, respectively. Additionally, assume some classifier exists, denoted by `model`, that allows for the prediction of confidence score vectors with a `predict` method.

Consider the following scenario. We would like to calibrate the predicted confidence score vectors on the test set using isotonic regression. Afterward, we would like to evaluate how well this post-hoc calibration method performs regarding confidence calibration by inspecting the conf-ECE_2 value. In order to perform these actions, we can use the following code snippet.

```
from uncertainty_estimation.calibrators import IsotonicRegression
from uncertainty_estimation.metrics import ExpectedCalibrationError

# obtain the predicted probabilities on the validation and test set
y_probs_val = model.predict(x_val)
y_probs_test = model.predict(x_test)

# instantiate the post-hoc calibrator and fit it on the validation set
# note that isotonic regression takes probabilities as input, not logits
calibrator = IsotonicRegression()
calibrator.fit(y_probs_val, y_val)

# re-calibrate the predicted probabilities on the test set
y_probs_test = calibrator.predict_proba(y_probs_test)

# evaluate confidence calibration using the conf-ECE_2
ece = ExpectedCalibrationError(y_probs_test, y_test)
conf_ece = ece.get_metric(ece_method='confidence',
                          bin_method='equal_range',
                          n_bins=15,
                          lp_norm=2)
```

7.2 Evaluation of calibration methods

With the previously introduced library, we now focus on evaluating different calibration methods on a variety of datasets. The objective of this evaluation is to study how different techniques that improve calibration compare to one another. For this evaluation, we consider ResNet-8, ResNet-14, ResNet-32, ResNet-50, ResNet-110, and ResNet-152. The details of these classifiers are covered in Appendix A.2. Briefly, the number in the classifier name indicates the number of layers. A high number of layers indicates that the classifier is more powerful, which implies that the risk of overfitting increases. We train these classifiers with the Brier score on CIFAR-10, CIFAR-100, and ImageWoof. For each training procedure, we select the classifier from the epoch in which it achieved the highest accuracy on the validation set. This selected classifier will be used for the evaluation. We then train all post-hoc calibration methods on the validation

set described in Table A.1. After training the post-hoc calibration methods, we generate predictions on the test set with the trained classifiers. After generating these predictions, they are fed into the trained post-hoc calibration methods to obtain re-calibrated predictions. Then, we evaluate the re-calibrated predictions based on their accuracy, conf-ECE_2, and CW-ECE_2. In this work, we do not study how the size of the validation set impacts the quality of the post-hoc calibration method. However, this direction would be interesting future work.

Initially, we performed the evaluation for classifiers trained *without* data augmentation, as data augmentation is one of the possible remedies for miscalibration. However, we found that data augmentation increases the accuracy attained by the classifiers substantially. Since classifiers are typically chosen based on accuracy, a practitioner would generally apply data augmentation. Hence, in order to consider a more realistic setting, it makes more sense to focus on calibration methods for classifiers trained *with* data augmentation. For completeness, we include the evaluation of the calibration methods applied to classifiers trained *without* data augmentation in Appendix D. We want to remark that data augmentation is already a preventative technique against miscalibration. Hence, by focusing on classifiers trained with data augmentation, there is less of an improvement to gain regarding calibration with post-hoc calibration methods.

7.2.1 Confidence calibration

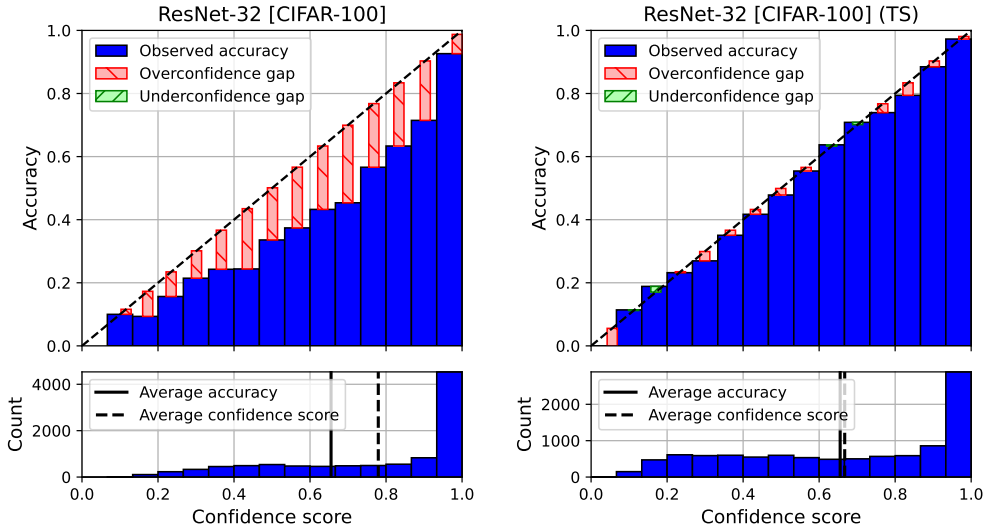
The conf-ECE_2 for multiple classifiers on several datasets is depicted in Table 7.1. For the uncalibrated classifier (denoted by "uncalibrated"), the table cells indicate the conf-ECE_2, together with the accuracy of the classifier. The table cells of all calibration methods indicate the conf-ECE_2 together with the percentage point increase or decrease in accuracy compared to the uncalibrated classifier. For instance, if the uncalibrated classifier achieves an accuracy of 0.90 and the accuracy becomes 0.88 after a post-hoc calibration method, then this decrease in accuracy is denoted by -0.2%. For each row, the **bold** table cell indicates the best performing technique concerning the conf-ECE_2 and the underlined table cell indicates the best performing technique concerning accuracy. All calibration methods have been abbreviated. From left to right, these are Temperature Scaling (TS), Platt Scaling (PS), Isotonic Regression (IR), Histogram Binning (HB), Beta Calibration (BC), Vector Scaling (VS), Matrix Scaling (MS), and Classifier Selection Strategy (CSS). We use 15 equally-ranged bins for histogram binning and the classifier selection strategy technique selects the classifier with the lowest loss on the validation set.

Table 7.1: The conf-ECE_2 for multiple classifiers trained with the Brier score on several datasets, *with* data augmentation. The values in the cells indicate the conf-ECE_2 together with the percent point increase or decrease in accuracy compared to the uncalibrated classifier in parenthesis. **Bold** values indicate the best performing techniques concerning the conf-ECE_2 and underlined values indicate the best performing techniques concerning the accuracy.

Dataset	Classifier	Mitigative								Preventative
		Accuracy preserving		Non-accuracy preserving						
		Uncalibrated	TS	PS	IR	HB	BC	VS	MS	CSS
CIFAR-10	ResNet-8	0.031 (86.4)	**0.014 (0.0)**	0.161 (-0.02)	0.019 (0.22)	0.032 (0.15)	0.017 (0.38)	0.017 (0.55)	0.016 (0.62)	0.026 (-0.02)
CIFAR-10	ResNet-14	0.061 (89.7)	0.021 (0.0)	0.11 (-0.29)	0.029 (-0.04)	0.045 (0.02)	0.02 (-0.02)	**0.017 (0.04)**	0.021 (-0.02)	0.058 (-0.01)
CIFAR-10	ResNet-32	0.062 (92.0)	0.019 (0.0)	0.053 (-0.13)	0.022 (-0.1)	0.037 (-0.4)	0.017 (-0.04)	**0.016 (-0.01)**	0.019 (-0.14)	0.062 (0.0)
CIFAR-10	ResNet-50	0.062 (92.3)	**0.019 (0.0)**	0.033 (0.13)	0.025 (0.01)	0.047 (-0.09)	0.021 (-0.02)	0.02 (0.05)	0.022 (-0.15)	0.062 (0.0)
CIFAR-10	ResNet-110	0.062 (92.9)	**0.021 (0.0)**	0.031 (0.05)	0.027 (0.08)	0.044 (-0.1)	0.022 (0.11)	**0.021 (0.07)**	0.024 (-0.17)	0.061 (-0.1)
CIFAR-10	ResNet-152	0.061 (93.3)	**0.022 (0.0)**	0.024 (0.04)	0.021 (-0.15)	0.044 (-0.34)	0.03 (-0.06)	0.026 (0.09)	0.03 (-0.11)	0.061 (0.0)
CIFAR-100	ResNet-8	0.056 (54.8)	0.03 (0.0)	0.277 (-1.74)	0.026 (-0.02)	0.091 (-3.03)	**0.017 (0.73)**	0.026 (0.76)	0.213 (-8.58)	0.056 (0.0)
CIFAR-100	ResNet-14	0.101 (62.3)	**0.014 (0.0)**	0.284 (-1.55)	0.055 (-0.31)	0.102 (-2.88)	0.036 (0.07)	0.02 (0.06)	0.254 (-9.72)	0.103 (-0.19)
CIFAR-100	ResNet-32	0.141 (65.5)	**0.018 (0.0)**	0.239 (-0.74)	0.062 (-0.34)	0.12 (-2.72)	0.044 (0.23)	0.03 (0.19)	0.236 (-9.0)	0.085 (-0.51)
CIFAR-100	ResNet-50	0.155 (67.2)	**0.027 (0.0)**	0.214 (-0.4)	0.067 (-0.39)	0.118 (-2.12)	0.055 (0.05)	0.036 (0.02)	0.242 (-9.08)	0.104 (-0.8)
CIFAR-100	ResNet-110	0.165 (68.8)	**0.034 (0.0)**	0.181 (-0.19)	0.074 (-0.36)	0.126 (-2.89)	0.062 (0.05)	0.042 (-0.19)	0.253 (-8.94)	0.092 (-1.08)
CIFAR-100	ResNet-152	0.162 (69.4)	**0.03 (0.0)**	0.178 (-0.11)	0.066 (-0.07)	0.122 (-2.5)	0.053 (0.24)	0.036 (-0.04)	0.242 (-8.45)	0.091 (-0.97)
ImageWoof	ResNet-8	0.025 (73.1)	0.033 (0.0)	0.252 (-0.31)	0.036 (-0.38)	0.048 (-0.99)	**0.017 (-0.38)**	0.032 (0.15)	0.026 (-0.27)	0.019 (-1.0)
ImageWoof	ResNet-14	0.056 (79.7)	0.038 (0.0)	0.208 (-0.92)	0.042 (0.08)	0.064 (-1.11)	0.045 (0.42)	0.034 (-0.23)	**0.027 (-0.42)**	0.061 (-0.54)
ImageWoof	ResNet-32	0.088 (81.6)	0.037 (0.0)	0.172 (-0.31)	0.051 (-0.80)	0.082 (-0.65)	**0.023 (-0.27)**	0.037 (-0.19)	0.029 (-0.61)	0.081 (0.04)
ImageWoof	ResNet-50	0.093 (82.0)	0.033 (0.0)	0.159 (-0.27)	0.042 (-0.46)	0.096 (-1.64)	0.028 (-0.27)	**0.024 (-0.19)**	0.03 (-0.34)	0.093 (0.0)
ImageWoof	ResNet-110	0.095 (82.3)	0.029 (0.0)	0.153 (-0.31)	0.046 (0.04)	0.087 (-0.80)	**0.022 (0.34)**	0.023 (-0.08)	0.032 (0.04)	0.092 (-0.07)
ImageWoof	ResNet-152	0.075 (81.9)	0.043 (0.0)	0.173 (0.23)	0.044 (0.73)	0.08 (-0.08)	**0.036 (0.19)**	0.048 (0.80)	0.038 (0.31)	0.075 (0.0)

The table shows that the conf-ECE_2 generally improves compared to the uncalibrated classifier by applying a calibration method. Up till a certain point, the degree of these improvements seems to increase with the classifier complexity. For instance, consider ResNet-8, ResNet-14, and ResNet-32 on CIFAR-100 when applying temperature scaling. For ResNet-8, the improvement in conf-ECE_2 is $(0.056 - 0.03)/0.056 \cdot 100 = 46.43\%$. This improvement is 86.14% for ResNet-14 and 87.23% for ResNet-32. Since such a massive improvement in conf-ECE_2 might be challenging to interpret intuitively, we include reliability diagrams of ResNet-32 with and without temperature scaling in Figure 7.1. This figure shows the substantial improvement gained by using a post-hoc calibration method. Furthermore, this post-hoc calibration method preserves the accuracy of the original classifier. In the density diagrams, one can observe that the overconfidence of the original classifier is tackled by primarily decreasing (high) confidence scores (*e.g.*, the rightmost bin goes from roughly 4000 samples in the uncalibrated setting to 2500 samples in the re-calibrated setting). Table 7.1 also

shows that selecting the classifier based on the lowest loss on the validation set instead of the highest accuracy on the test set (*i.e.*, the CSS method) improves the calibration for several classifiers on CIFAR-100. Specifically, the conf-ECE_2 improves significantly for ResNet-32, Resnet-50, ResNet-110, and ResNet-152.



(a) Without temperature scaling **(b)** With temperature scaling

Figure 7.1: Reliability diagrams for (a) the original ResNet-32 classifier on CIFAR-100 and (b) the classifier that uses temperature scaling as post-hoc calibration method.

Across all methods in Table 7.1, we observe that no calibration method consistently outperforms all others. We do, however, observe that primarily temperature scaling, vector scaling, and beta calibration tend to work well across all classifiers and datasets. These latter two techniques do generally come paired with an increase or decrease in accuracy. In a practical setting, however, it might be favorable to guarantee that the calibration method does not modify the ranking of the top-1 confidence scores, thereby preserving the accuracy of the original classifier. Since many classifiers are evaluated based on their accuracy, guaranteeing that the post-hoc calibration method does not impact the accuracy makes it more favorable for practical use. Furthermore, if the calibration method can change the confidence score ranking, it gets more sensitive to what degree the calibration set represents the data seen in the real world. For instance, if the calibration set is not representative of real-world data, a calibration method that can modify the confidence score ranking might lead to a grave decrease in

accuracy. Such a decrease is highly undesirable. Due to this accuracy-preserving property, temperature scaling seems a good choice for a calibration method. This observation is interesting as temperature scaling is the most straightforward post-hoc calibration method as it only requires learning a single parameter. Another noteworthy observation is that matrix scaling decreases the accuracy substantially for all classifiers on CIFAR-100. We believe this behavior stems from matrix scaling overfitting to the validation set. We speculate that this is particularly noticeable on CIFAR-100 due to the number of parameters for matrix scaling growing quadratically in the number of classes. Hence, matrix scaling contains substantially more parameters for CIFAR-100 than CIFAR-10 and ImageWoof, which might lead to overfitting. The main takeaway of this section is that post-hoc calibration can substantially improve the calibration of a classifier under confidence calibration with relatively straightforward techniques.

7.2.2 Class-wise calibration

In the previous section, we focused on confidence calibration by studying the conf-ECE$_2$. In this section, we study class-wise calibration in the form of the CW-ECE$_2$. Table 7.2 depicts the CW-ECE$_2$ across the classifiers and datasets. We multiplied the CW-ECE$_2$ values by 10^2 in these cells to reduce the number of decimals zeroes in the front. Next to the CW-ECE$_2 \cdot 10^2$, these cells also contain the percentage point increase or decrease in accuracy compared to the uncalibrated classifier.

Table 7.2: The CW-ECE$_2$ for multiple classifiers trained with the Brier score on several datasets, *with* data augmentation. The values in the cells indicate the CW-ECE$_2 \cdot 10^2$ together with the percentage point increase or decrease in accuracy compared to the uncalibrated classifier in parenthesis. **Bold** values indicate the best performing techniques concerning the CW-ECE$_2$ and underlined values indicate the best performing techniques concerning the accuracy.

Dataset	Classifier	Mitigative								Preventative
		Accuracy preserving		Non-accuracy preserving						
		Uncalibrated	TS	PS	IR	HB	BC	VS	MS	CSS
CIFAR-10	ResNet-8	2.262 (86.4)	2.099 (0.0)	6.148 (-0.02)	2.002 (0.22)	2.197 (0.15)	1.724 (0.38)	1.653 (0.55)	**1.51 (0.62)**	2.178 (-0.02)
CIFAR-10	ResNet-14	2.65 (89.7)	1.888 (0.0)	4.497 (-0.29)	2.061 (-0.04)	2.472 (0.02)	1.702 (-0.02)	1.767 (0.04)	**1.673 (-0.02)**	2.578 (-0.01)
CIFAR-10	ResNet-32	2.564 (92.0)	**1.619 (0.0)**	2.786 (-0.13)	1.817 (-0.1)	1.968 (-0.4)	1.699 (-0.04)	1.648 (-0.01)	1.756 (-0.14)	2.564 (0.0)
CIFAR-10	ResNet-50	2.691 (92.3)	1.849 (0.0)	2.068 (0.13)	1.9 (0.01)	2.393 (-0.09)	**1.772 (-0.02)**	1.872 (0.05)	1.826 (-0.15)	2.691 (0.0)
CIFAR-10	ResNet-110	2.917 (92.9)	2.115 (0.0)	2.037 (0.05)	1.964 (0.08)	2.442 (-0.1)	1.891 (0.11)	1.695 (0.07)	**1.666 (-0.17)**	2.717 (-0.1)
CIFAR-10	ResNet-152	2.798 (93.3)	2.014 (0.0)	1.886 (0.04)	2.059 (-0.15)	2.261 (-0.34)	2.041 (-0.06)	1.906 (0.09)	**1.872 (-0.11)**	2.798 (0.0)
CIFAR-100	ResNet-8	1.878 (54.8)	1.782 (0.0)	2.955 (-1.74)	1.748 (-0.02)	1.771 (-3.03)	1.647 (0.73)	**1.598 (0.76)**	2.699 (-8.58)	1.878 (0.0)
CIFAR-100	ResNet-14	2.016 (62.3)	1.728 (0.0)	2.964 (-1.55)	1.78 (-0.31)	1.879 (-2.88)	1.705 (0.07)	**1.644 (0.06)**	3.06 (-9.72)	2.012 (-0.19)
CIFAR-100	ResNet-32	2.157 (65.5)	1.681 (0.0)	2.664 (-0.74)	1.799 (-0.34)	1.991 (-2.72)	1.714 (0.23)	**1.674 (0.19)**	2.904 (-9.0)	1.930 (-0.51)
CIFAR-100	ResNet-50	2.311 (67.2)	1.738 (0.0)	2.438 (-0.4)	1.835 (-0.39)	1.94 (-2.12)	1.793 (0.05)	**1.686 (0.02)**	2.958 (-9.08)	2.060 (-0.8)
CIFAR-100	ResNet-110	2.338 (68.8)	1.735 (0.0)	2.244 (-0.19)	1.81 (-0.36)	1.969 (-2.89)	1.764 (0.05)	**1.685 (-0.19)**	3.036 (-8.94)	1.937 (-1.08)
CIFAR-100	ResNet-152	2.287 (69.4)	**1.719 (0.0)**	2.268 (-0.11)	1.758 (-0.07)	2.008 (-2.5)	1.755 (0.24)	1.744 (-0.04)	2.964 (-8.45)	1.899 (-0.97)
ImageWoof	ResNet-8	3.214 (73.1)	**3.003 (0.0)**	8.451 (-0.31)	3.782 (-0.38)	3.543 (-0.99)	3.38 (-0.38)	3.187 (0.15)	3.05 (-0.27)	3.017 (-1.0)
ImageWoof	ResNet-14	3.901 (79.7)	3.642 (0.0)	7.28 (-0.92)	3.706 (0.08)	4.352 (-1.11)	3.586 (0.42)	3.127 (-0.23)	**3.058 (-0.42)**	3.864 (-0.54)
ImageWoof	ResNet-32	4.316 (81.6)	3.37 (0.0)	6.523 (-0.31)	3.543 (-0.80)	4.442 (-0.65)	**3.18 (-0.27)**	3.435 (-0.19)	3.253 (-0.61)	4.336 (0.04)
ImageWoof	ResNet-50	4.604 (82.0)	3.512 (0.0)	6.164 (-0.27)	3.724 (-0.46)	4.529 (-1.64)	3.412 (-0.27)	3.58 (-0.19)	**3.249 (-0.34)**	4.245 (0.0)
ImageWoof	ResNet-110	4.278 (82.3)	3.471 (0.0)	6.113 (-0.31)	3.662 (0.04)	3.843 (-0.80)	**3.092 (0.34)**	3.413 (-0.08)	3.194 (0.04)	4.363 (-0.07)
ImageWoof	ResNet-152	4.076 (81.9)	3.502 (0.0)	6.435 (0.23)	3.481 (0.73)	4.251 (-0.08)	3.42 (0.19)	**3.277 (0.80)**	3.588 (0.31)	4.076 (0.0)

From the table, we observe that most calibration methods improve the CW-ECE$_2$ compared to the uncalibrated classifier. This table shows that matrix scaling performs relatively well concerning calibration on CIFAR-10 and ImageWoof but substantially worse on CIFAR-100. Similar to the previous section, we again speculate that the reason for matrix scaling performing worse on CIFAR-100 is due to overfitting on the validation set. We also observe that vector scaling tends to perform relatively well compared to other techniques. While temperature scaling seemed to (slightly) outperform vector scaling concerning the conf-ECE$_2$, vector scaling seems to outperform temperature scaling concerning the CW-ECE$_2$. We can explain vector scaling outperforming temperature scaling concerning the CW-ECE$_2$ based on vector scaling being able to learn different behavior for different classes. Temperature scaling cannot learn different behavior as it only contains a single parameter. We also observed that temperature scaling generally outperforms vector scaling concerning the conf-ECE$_2$. Intuitively, this implies that vector scaling performs better for classes that are

not the predicted class. Hence, vector scaling might be better at calibrating lower confidence scores. The main takeaway of this section is that post-hoc calibration can generally substantially improve the class-wise calibration of a classifier.

7.3 Calibration for classifiers trained using the log-loss

We repeat the experiments in the previous section for a variety of classifiers trained using the log-loss instead of the Brier score[1]. This section shows that the effectiveness of calibration methods is not limited to classifiers trained with the Brier score. Table 7.3 depicts the conf-ECE_2 for these classifiers, together with a variety of calibration methods. We exclude the classifier selection strategy technique as we do not have access to all classifiers generated throughout the epochs. Similar to § 7.2.1, we observe that temperature scaling, vector scaling, and matrix scaling show promising results. Despite matrix scaling showing promising results concerning calibration, the accuracy still drops substantially for CIFAR-100. These results are in line with the findings in § 7.2.1. Table 7.4 depicts the CW-ECE_2 for the classifiers. Similar to § 7.2.2, we again observe that temperature scaling, vector scaling, and matrix scaling perform well. These findings align with those found earlier.

[1]Excluding LeNet-5, these trained classifiers have been obtained from https://github.com/markus93/NN_calibration.

Table 7.3: The conf-ECE_2 for multiple classifiers trained with the log-loss on several datasets, *with* data augmentation. The values in the cells indicate the conf-ECE_2 together with the percentage point increase or decrease in accuracy compared to the uncalibrated classifier in parenthesis. **Bold** values indicate the best performing techniques concerning the conf-ECE_2 and underlined values indicate the best performing techniques concerning the accuracy.

Dataset	Classifier	Mitigative							
		Accuracy preserving		Non-accuracy preserving					
		UC	TS	PS	IR	HB	BC	VS	MS
CIFAR-10	LeNet-5	0.033 (73.9)	0.015 (0.0)	0.217 (0.21)	0.02 (0.65)	0.028 (0.5)	0.026 (0.6)	**0.011 (0.89)**	0.013 (0.83)
CIFAR-10	ResNet-110	0.066 (93.6)	0.025 (0.0)	0.028 (-0.02)	0.025 (0.06)	0.041 (-0.1)	0.032 (0.0)	**0.022 (-0.01)**	0.027 (-0.23)
CIFAR-10	ResNet-110 (SD)	0.059 (92.9)	0.018 (0.0)	0.126 (-0.03)	0.023 (0.05)	0.034 (-0.12)	0.021 (0.28)	0.017 (0.34)	**0.016 (0.15)**
CIFAR-10	Wide ResNet 16 8	0.051 (94.9)	**0.021 (0.0)**	0.022 (-0.02)	0.022 (-0.15)	0.045 (-0.24)	0.028 (0.06)	0.026 (0.1)	**0.021 (-0.11)**
CIFAR-10	DenseNet-40	0.082 (92.4)	**0.016 (0.0)**	0.107 (-0.22)	0.026 (-0.07)	0.059 (-0.25)	0.034 (-0.01)	0.022 (0.06)	0.022 (-0.15)
CIFAR-100	LeNet-5	0.056 (38.5)	0.023 (0.0)	0.205 (-1.48)	0.047 (0.1)	0.066 (-1.8)	0.022 (1.0)	**0.018 (1.15)**	0.217 (-6.13)
CIFAR-100	ResNet-110	0.209 (71.5)	**0.027 (0.0)**	0.18 (-0.36)	0.072 (-0.22)	0.116 (-1.37)	0.065 (0.18)	0.032 (-0.04)	0.278 (-10.43)
CIFAR-100	ResNet-110 (SD)	0.163 (73.4)	**0.014 (0.0)**	0.316 (-2.56)	0.052 (0.33)	0.116 (-1.94)	0.045 (0.42)	0.049 (0.27)	0.282 (-13.54)
CIFAR-100	Wide ResNet 16 8	0.098 (76.6)	**0.047 (0.0)**	0.165 (-0.17)	0.064 (-0.27)	0.112 (-2.1)	0.055 (0.12)	0.054 (0.13)	0.243 (-8.85)
CIFAR-100	DenseNet-40	0.233 (70.0)	**0.012 (0.0)**	0.3 (-1.96)	0.061 (0.04)	0.1 (-1.21)	0.075 (0.19)	0.045 (0.13)	0.353 (-15.52)

Table 7.4: The CW-ECE_2 for multiple classifiers trained with the log-loss on several datasets, *with* data augmentation. The values in the cells indicate the CW-$ECE_2 \cdot 10^2$ together with the percentage point increase or decrease in accuracy compared to the uncalibrated classifier in parenthesis. **Bold** values indicate the best performing techniques concerning the CW-ECE_2 and underlined values indicate the best performing techniques concerning the accuracy.

Dataset	Classifier	Mitigative							
		Accuracy preserving		Non-accuracy preserving					
		UC	TS	PS	IR	HB	BC	VS	MS
CIFAR-10	LeNet-5	2.616 (73.9)	2.618 (0.0)	7.305 (0.21)	2.205 (0.65)	2.12 (0.5)	2.145 (0.6)	**1.612 (0.89)**	1.798 (0.83)
CIFAR-10	ResNet-110	2.825 (93.6)	**1.764 (0.0)**	1.816 (-0.02)	2.02 (0.06)	2.274 (-0.1)	2.001 (0.0)	**1.764 (-0.01)**	2.085 (-0.23)
CIFAR-10	ResNet-110 (SD)	2.822 (92.9)	1.931 (0.0)	5.041 (-0.03)	2.028 (0.05)	2.018 (-0.12)	1.91 (0.28)	1.714 (0.34)	**1.594 (0.15)**
CIFAR-10	Wide ResNet 16 8	2.618 (94.9)	1.85 (0.0)	1.807 (-0.02)	2.019 (-0.15)	2.164 (-0.24)	2.032 (0.06)	**1.675 (0.1)**	1.84 (-0.11)
CIFAR-10	DenseNet-40	3.482 (92.4)	**1.692 (0.0)**	4.516 (-0.22)	2.022 (-0.07)	2.624 (-0.25)	1.881 (-0.01)	1.737 (0.06)	1.953 (-0.15)
CIFAR-100	LeNet-5	1.839 (38.5)	1.701 (0.0)	2.344 (-1.48)	1.727 (0.1)	1.667 (-1.8)	1.664 (1.0)	**1.566 (1.15)**	2.68 (-6.13)
CIFAR-100	ResNet-110	2.659 (71.5)	**1.662 (0.0)**	2.265 (-0.36)	1.823 (-0.22)	1.993 (-1.37)	1.767 (0.18)	1.767 (-0.04)	3.267 (-10.43)
CIFAR-100	ResNet-110 (SD)	2.315 (73.4)	**1.663 (0.0)**	3.307 (-2.56)	1.733 (0.33)	1.958 (-1.94)	1.699 (0.42)	1.699 (0.27)	3.364 (-13.54)
CIFAR-100	Wide ResNet 16 8	1.951 (76.6)	1.793 (0.0)	2.09 (-0.17)	**1.723 (-0.27)**	1.905 (-2.1)	1.763 (0.12)	1.738 (0.13)	2.986 (-8.85)
CIFAR-100	DenseNet-40	2.957 (70.0)	**1.623 (0.0)**	3.14 (-1.96)	1.771 (0.04)	1.93 (-1.21)	1.807 (0.19)	1.713 (0.13)	4.008 (-15.52)

7.4 Discussion

The evaluation in the previous sections shows that most calibration methods improve calibration. Especially temperature scaling, vector scaling, and matrix scaling seem effective methods for improving calibration. In our experiments, temperature scaling and vector scaling work best for confidence calibration, whereas vector scaling and matrix scaling work best for class-wise calibration. However, temperature scaling should not be dismissed for class-wise calibration due to its straightforwardness and accuracy-preserving nature. These properties render this method promising despite it potentially being slightly less effective than other methods. Although matrix scaling provides substantial improvements regarding calibration, it also led to a significant decrease in accuracy (ranging between 8.45% and 9.72%) for classifiers trained with the Brier score on CIFAR-100. This property alone can make this method undesirable for practical settings. Based on these observations, we believe both temperature scaling and vector scaling to be good options for calibration methods. Whether one should opt for temperature scaling or vector scaling depends on the desirability of the accuracy-preserving property. If one would like the accuracy of the classifier to be preserved, one might prefer temperature scaling over vector scaling. The primary disadvantage of temperature scaling boils down to being unable to learn different behavior for different classes. Consequently, this method might be outperformed by vector scaling for class-wise calibration. Based on this remark, a promising future direction arises. This direction consists of designing a post-hoc calibration method that can learn different behavior for different classes without impacting the ranking of the top-1 confidence score. Under this constraint, the method will preserve the accuracy of the classifier while allowing for more flexibility. For such a method, one might be interested in exploring strictly isotonic functions as these preserve the ordering of confidence scores.

Chapter 8

Conclusion

In this work, we discussed the calibration of classifiers. We started by introducing the concept of calibration under three different notions in Chapter 2. Generally, calibration ensures that predicted confidence scores properly reflect empirical frequencies. Based on various examples, we argued that calibration is a desirable property for classifiers in many classification tasks. In order to reason about the calibration of a classifier, we introduced different approaches for evaluating calibration in Chapter 3.

In Chapter 4, we found that any loss function that is a proper scoring rule can be decomposed into a calibration, refinement, and dataset loss. Therefore, by minimizing the calibration loss during training, we hypothesized that the calibration of classifiers would improve throughout the training procedure. Empirically, however, we found that classifiers tend to be miscalibrated. Based on this observation, we proposed two hypotheses for the origination of miscalibration, both rooted in (slight) differences between the training and test set. We hypothesized that overfitting to the loss function results in a gap between the degree of calibration on the training and test set. In this case, the classifier also achieved a substantially higher accuracy on the training set than on the test set. Based on the training set, the classifier learned a certain degree of uncertainty that is representative of its accuracy. However, since the classifier achieves a substantially lower accuracy on the test set than the training set, this learned uncertainty does not accurately represent the accuracy achieved on the test set. Consequently, this classifier generally produces overconfident predictions on the test set. We verified these hypotheses with an empirical evaluation and found support for them to hold. Based on this knowledge, we introduced various calibration methods that improve the calibration of classifiers either during training (Chapter 5) or after training (Chapter 6). Afterward, we studied the effectiveness of these calibration methods on several classifiers across different datasets in Chapter 7. In order to perform these evaluations, we designed and implemented a Python library that contains all post-hoc calibration methods proposed in this work. Furthermore, this library allows for the evaluation of any classifier regard-

ing both confidence and class-wise calibration through the conf-ECE, CW-ECE, confidence reliability diagrams, and class-wise reliability diagrams. From the experimental results, we observed that the calibration methods seem effective for improving the calibration of a classifier. In particular, temperature scaling and vector scaling substantially reduced the degree of miscalibration across many classifiers and datasets.

8.1 Limitations

In this section, we briefly cover some limitations of the findings in this work. A primary limitation of this work concerns the experimental evaluation for verifying the overfitting and distribution shift hypothesis in § 4.6. This evaluation was performed for a collection of classifiers with the same general architecture (*i.e.*, ResNet), all trained with the Brier score. In order to find even more support for these hypotheses, one could perform equivalent experiments for different classifier architectures. Another difficulty we ran into is the estimation of the calibration loss under the log-loss. Due to this limitation, we resorted to verifying the hypotheses using the conf-ECE_2 and CW-ECE_2 instead of the CL^{LL}. It might still be feasible to come to an accurate estimation of the CL^{LL} by using kernel density estimation instead of binning.

Another limitation lies in the evaluation of the preventative calibration methods for miscalibration. We introduced four different preventative techniques (data augmentation, changing the performance criterion, label smoothing, and modifying the loss function). However, many more preventative techniques exist (*e.g.*, any technique that reduces overfitting). Furthermore, due to time constraints, we only focused on two of these techniques (data augmentation and changing the performance criterion) in the experimental evaluation. Therefore, there is still a lot of room for improvement concerning the evaluation of preventative techniques.

The focus on a specific type of classifiers and tasks results in another limitation. Since we only focused on computer vision tasks, we are unaware of whether our findings transfer to other domains, such as classification in natural language processing. Furthermore, we are unaware whether our findings transfer to different types of classifiers, such as decision trees and support vector machines.

8.2 Future work

In this section, we provide an overview of possible future research directions. Due to the effectiveness of post-hoc calibration methods, most research directions are related to improving post-hoc calibration.

Calibration set size In order to study the effectiveness of different post-hoc calibration methods, we selected a calibration set of fixed size. A future research direction resides in studying the impact of different calibration set sizes. Preferably, the size of this set is minimized to ensure the training set consists of as many samples as possible. Then, an interesting direction would be to study the dynamic between the number of samples in the calibration set and the performance of the post-hoc calibration methods.

Accuracy-preserving post-hoc calibration Temperature scaling seems attractive thanks to its accuracy-preserving property. However, it is unable to learn different behavior for different classes. This limitation might restrict its effectiveness for class-wise calibration in various cases. Vector scaling would be able to learn more complex relationships. However, this method does not necessarily preserve the accuracy of the original classifier. Therefore, an interesting future direction could be designing a post-hoc calibration method that allows for more complex relationships between logits and re-calibrated confidence score vectors than temperature scaling and still preserves the accuracy. For this, one could investigate strictly isotonic functions.

Calibration and out-of-distribution samples Post-hoc calibration methods are trained with a calibration set. Typically, such a calibration set consists of in-distribution samples. Based on this observation, we wonder how post-hoc calibration performs on out-of-distribution samples. Specifically, we wonder whether post-hoc calibration methods allow for the detection of out-of-distribution samples. If not, it would be interesting to study whether adding out-of-distribution samples with a uniformly distributed ground truth vector to the calibration set helps.

Calibration metrics The calibration metrics introduced in this work are not ideal for measuring the degree of miscalibration due to binning or other grouping procedures (such as kernel density estimation), which introduces estimation errors. Widmann et al. [51] introduced a suite of statistical hypothesis tests whose null hypothesis is that the classifier is well-calibrated. The evaluation in their

work focused primarily on synthetic datasets and classifiers. Therefore, it would be interesting to use their statistical tests on different classifier architectures on more realistic datasets. One could then compare the conclusion of their test with reliability diagrams (and other tools) to derive whether these tests are a helpful addition to the toolkit of calibration evaluation.

Preventative calibration methods for miscalibration In this work, we found support that overfitting is an important cause of miscalibration. Based on this observation, we introduced and studied preventative (and mitigative) calibration methods. As argued in the previous section, we only considered two preventative techniques while studying the effectiveness of preventative techniques. Hence, there is still much room for future research. For instance, one could test whether different techniques that reduce overfitting help improve the calibration and, if so, to what degree. Furthermore, it would be interesting to investigate whether a single mitigative technique can almost always outperform one or more preventative techniques. That is, can we ignore preventative techniques and instead solely focus on mitigative techniques?

Bibliography

[1] Bejani, M. M. and Ghatee, M. (2021). A systematic review on overfitting control in shallow and deep neural networks. *Artificial Intelligence Review*, pages 1–48.

[2] Böken, B. (2021). On the appropriateness of Platt scaling in classifier calibration. *Information Systems*, 95:101641.

[3] Brier, G. W. et al. (1950). Verification of forecasts expressed in terms of probability. *Monthly weather review*, 78(1):1–3.

[4] Chen, B., Ziyin, L., Wang, Z., and Liang, P. P. (2020). An investigation of how label smoothing affects generalization. *arXiv preprint arXiv:2010.12648*.

[5] DeGroot, M. H. and Fienberg, S. E. (1983). The comparison and evaluation of forecasters. *Journal of the Royal Statistical Society: Series D (The Statistician)*, 32(1-2):12–22.

[6] Deng, J., Dong, W., Socher, R., Li, L., Li, K., and Li, F. (2009). 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*, pages 248–255. IEEE Computer Society.

[7] Everitt, B. S. and Skrondal, A. (2010). The Cambridge dictionary of statistics.

[8] Gneiting, T. and Raftery, A. E. (2007). Strictly proper scoring rules, prediction, and estimation. *Journal of the American statistical Association*, 102(477):359–378.

[9] Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. (2017). On calibration of modern neural networks. In Precup, D. and Teh, Y. W., editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 1321–1330. PMLR.

[10] Gupta, C. and Ramdas, A. K. (2021). Top-label calibration. *arXiv preprint arXiv:2107.08353*.

[11] Gupta, K., Rahimi, A., Ajanthan, T., Mensink, T., Sminchisescu, C., and Hartley, R. (2020). Calibration of neural networks using splines. *arXiv preprint arXiv:2006.12800*.

[12] He, K., Zhang, X., Ren, S., and Sun, J. (2016). 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*, pages 770–778. IEEE Computer Society.

[13] Heiser, T. J. T., Allikivi, M.-L., and Kull, M. (2019). Shift Happens: Adjusting Classifiers. In *ECML/PKDD (2)*.

[14] Hendrycks, D., Lee, K., and Mazeika, M. (2019). Using pre-training can improve model robustness and uncertainty. In Chaudhuri, K. and Salakhutdinov, R., editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 2712–2721. PMLR.

[15] Huang, G., Liu, Z., van der Maaten, L., and Weinberger, K. Q. (2017). Densely connected convolutional networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 2261–2269. IEEE Computer Society.

[16] Huang, G., Sun, Y., Liu, Z., Sedra, D., and Weinberger, K. Q. (2016). Deep networks with stochastic depth. In *European conference on computer vision*, pages 646–661. Springer.

[17] Karandikar, A., Cain, N., Tran, D., Lakshminarayanan, B., Shlens, J., Mozer, M. C., and Roelofs, R. (2021). Soft Calibration Objectives for Neural Networks. In *Thirty-Fifth Conference on Neural Information Processing Systems*.

[18] Kermany, D. S., Goldbaum, M., Cai, W., Valentim, C. C., Liang, H., Baxter, S. L., McKeown, A., Yang, G., Wu, X., Yan, F., et al. (2018). Identifying medical diagnoses and treatable diseases by image-based deep learning. *Cell*, 172(5):1122–1131.

[19] Kiefer, J. and Wolfowitz, J. (1952). Stochastic estimation of the maximum of a regression function. *The Annals of Mathematical Statistics*, pages 462–466.

[20] Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images.

[21] Kull, M., de Menezes e Silva Filho, T., and Flach, P. A. (2017). Beta calibration: a well-founded and easily implemented improvement on logistic calibration for binary classifiers. In Singh, A. and Zhu, X. J., editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA*, volume 54 of *Proceedings of Machine Learning Research*, pages 623–631. PMLR.

[22] Kull, M. and Flach, P. (2015). Novel decompositions of proper scoring rules for classification: Score adjustment as precursor to calibration. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 68–85. Springer.

[23] Kull, M., Perelló-Nieto, M., Kängsepp, M., de Menezes e Silva Filho, T., Song, H., and Flach, P. A. (2019). Beyond temperature scaling: Obtaining well-calibrated multi-class probabilities with dirichlet calibration. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 12295–12305.

[24] Kumar, A., Liang, P., and Ma, T. (2019). Verified uncertainty calibration. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 3787–3798.

[25] Laves, M.-H., Ihler, S., Kortmann, K.-P., and Ortmaier, T. (2019). Well-calibrated model uncertainty with temperature scaling for dropout variational inference. *arXiv preprint arXiv:1909.13550*.

[26] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

[27] Litjens, G., Kooi, T., Bejnordi, B. E., Setio, A. A. A., Ciompi, F., Ghafoorian, M., Van Der Laak, J. A., Van Ginneken, B., and Sánchez, C. I. (2017). A survey on deep learning in medical image analysis. *Medical image analysis*, 42:60–88.

[28] McLuckie, A. (2009). *Respiratory disease and its management*. Springer Science & Business Media.

[29] Merkle, E. C. and Steyvers, M. (2013). Choosing a strictly proper scoring rule. *Decision Analysis*, 10(4):292–304.

[30] Mikołajczyk, A. and Grochowski, M. (2018). Data augmentation for improving deep learning in image classification problem. In *2018 international interdisciplinary PhD workshop (IIPhDW)*, pages 117–122. IEEE.

[31] Mitchell, K. and Ferro, C. (2017). Proper scoring rules for interval probabilistic forecasts. *Quarterly Journal of the Royal Meteorological Society*, 143(704):1597–1607.

[32] Müller, R., Kornblith, S., and Hinton, G. E. (2019). When does label smoothing help? In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 4696–4705.

[33] Nabian, M. A. and Meidani, H. (2020). Physics-driven regularization of deep neural networks for enhanced engineering design and analysis. *Journal of Computing and Information Science in Engineering*, 20(1):011006.

[34] Niculescu-Mizil, A. and Caruana, R. (2005). Predicting good probabilities with supervised learning. In Raedt, L. D. and Wrobel, S., editors, *Machine Learning, Proceedings of the Twenty-Second International Conference (ICML 2005), Bonn, Germany, August 7-11, 2005*, volume 119 of *ACM International Conference Proceeding Series*, pages 625–632. ACM.

[35] Nixon, J., Dusenberry, M., Jerfel, G., Nguyen, T., Liu, J., Zhang, L., and Tran, D. (2019). Measuring Calibration in Deep Learning. *arXiv e-prints*, pages arXiv–1904.

[36] Northcutt, C. G., Athalye, A., and Mueller, J. (2021). Pervasive label errors in test sets destabilize machine learning benchmarks. *arXiv preprint arXiv:2103.14749*.

[37] Obadinma, S., Guo, H., and Zhu, X. (2021). Class-wise Calibration: A Case Study on COVID-19 Hate Speech. In *The 34th Canadian Conference on Artificial Intelligence, Vancouver*.

[38] Pereyra, G., Tucker, G., Chorowski, J., Kaiser, Ł., and Hinton, G. (2017). Regularizing neural networks by penalizing confident output distributions. *arXiv preprint arXiv:1701.06548*.

[39] Platt, J. et al. (1999). Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74.

[40] Posocco, N. and Bonnefoy, A. (2021). Estimating Expected Calibration Errors. In *International Conference on Artificial Neural Networks*, pages 139–150. Springer.

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

[42] Roelofs, R., Cain, N., Shlens, J., and Mozer, M. C. (2020). Mitigating bias in calibration error estimation. *arXiv preprint arXiv:2012.08668*.

[43] Rudan, I., Boschi-Pinto, C., Biloglav, Z., Mulholland, K., and Campbell, H. (2008). Epidemiology and etiology of childhood pneumonia. *Bulletin of the world health organization*, 86:408–416B.

[44] Savitzky, A. and Golay, M. J. (1964). Smoothing and differentiation of data by simplified least squares procedures. *Analytical chemistry*, 36(8):1627–1639.

[45] Shorten, C. and Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):1–48.

[46] Sreekumar, S. and Goldfeld, Z. (2021). Neural Estimation of Statistical Divergences. *arXiv preprint arXiv:2110.03652*.

[47] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 2818–2826. IEEE Computer Society.

[48] Vaicenavicius, J., Widmann, D., Andersson, C. R., Lindsten, F., Roll, J., and Schön, T. B. (2019). Evaluating model calibration in classification. In Chaudhuri, K. and Sugiyama, M., editors, *The 22nd International Conference on Artificial Intelligence and Statistics, AISTATS 2019, 16-18 April 2019, Naha, Okinawa, Japan*, volume 89 of *Proceedings of Machine Learning Research*, pages 3459–3467. PMLR.

[49] Weijs, S. V., Van Nooijen, R., and Van De Giesen, N. (2010). Kullback-Leibler divergence as a forecast skill score with classic reliability-resolution-uncertainty decomposition. *Monthly Weather Review*, 138(9):3387–3399.

[50] Wenger, J., Kjellström, H., and Triebel, R. (2020). Non-parametric calibration for classification. In Chiappa, S. and Calandra, R., editors, *The 23rd International Conference on Artificial Intelligence and Statistics, AISTATS 2020, 26-28 August 2020, Online [Palermo, Sicily, Italy]*, volume 108 of *Proceedings of Machine Learning Research*, pages 178–190. PMLR.

[51] Widmann, D., Lindsten, F., and Zachariah, D. (2019). Calibration tests in multi-class classification: A unifying framework. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 12236–12246.

[52] World Health Organisation (2021). Pneumonia. https://www.who.int/news-room/fact-sheets/detail/pneumonia, Last accessed on 21-11-2021.

[53] Ying, X. (2019). An overview of overfitting and its solutions. In *Journal of Physics: Conference Series*, volume 1168, page 022022. IOP Publishing.

[54] Zadrozny, B. and Elkan, C. (2001). Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers. In Brodley, C. E. and Danyluk, A. P., editors, *Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001), Williams College, Williamstown, MA, USA, June 28 - July 1, 2001*, pages 609–616. Morgan Kaufmann.

[55] Zadrozny, B. and Elkan, C. (2002). Transforming classifier scores into accurate multiclass probability estimates. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 694–699.

[56] Zagoruyko, S. and Komodakis, N. (2016). Wide residual networks. In Wilson, R. C., Hancock, E. R., and Smith, W. A. P., editors, *Proceedings of the British Machine Vision Conference 2016, BMVC 2016, York, UK, September 19-22, 2016*. BMVA Press.

[57] Zhang, J., Kailkhura, B., and Han, T. Y.-J. (2020). Mix-n-match: Ensemble and compositional methods for uncertainty calibration in deep learning. In *International Conference on Machine Learning*, pages 11117–11128. PMLR.

Appendices

Appendix A

Experimental setup

This chapter describes the experimental setup for the evaluations performed in this work. Specifically, it describes which datasets are used throughout our experiments, together with details on those datasets, such as the number of samples in the training, validation, and test set. Afterward, it introduces the model architectures relevant to this work. We also highlight the training procedure for the classifiers created based on these model architectures, including steps taken for data augmentation.

A.1 Datasets

The datasets relevant in our evaluations are CIFAR-10 [20], CIFAR-100 [20], and ImageWoof[1]. ImageWoof is a subset of ten classes from ImageNet [6], each class representing a different dog breed. Table A.1 presents details regarding each dataset, including the size of the training, validation, and test set. In addition to popular benchmark datasets CIFAR-10 and CIFAR-100, we include ImageWoof in some of our evaluations because we believe it to be more representative of a real-world use case. Its classes are relatively challenging to distinguish as they are all different kinds of dog breeds. Furthermore, as depicted in Table A.1, the images in this datasets are substantially larger than those in CIFAR-10 and CIFAR-100.

Table A.1: Details on the datasets used throughout this work.

Dataset	Image size	# classes	Training set	Validation set	Test set
CIFAR-10	32×32	10	45000	5000	10000
CIFAR-100	32×32	100	45000	5000	10000
ImageWoof	160×160	10	9027	1309	2620

[1]The ImageWoof dataset can be found at https://github.com/fastai/imagenette.

A.2 Architectures

Throughout this work, we use a variety of different neural network classifiers to ensure the stability of our results. Before listing all classifiers used throughout this work, we first note that each network is associated with a specific architecture. The classifiers we use are associated with either the ResNet [12], ResNet SD [16], DenseNet [15], Wide ResNet [56], or LeNet [26] architecture. For the experiments, we generally focus on the ResNet-8, ResNet-14, ResNet-32, ResNet-50, ResNet-110, and ResNet-152 classifiers, which originate from the ResNet architecture[2]. Throughout our work, we train these networks with the Brier score as the loss function. We sometimes also use some networks trained with the log-loss. These networks are LeNet-5, DenseNet-40, ResNet-110 SD, and Wide ResNet 16 8^3. Each number in the network name generally indicates the number of layers in the network. Hence, our collection of networks consist of a wide variety concerning model complexity. For details regarding the networks, we refer to the paper associated with each network architecture.

A.3 Training procedure

For all networks, we stick to the training procedure proposed in the paper associated with its respective architecture. For details, we refer the reader to our source code. Generally, this training procedure uses the Stochastic Gradient Descent (SGD) [19] optimizer with an accelerating Nesterov momentum of 0.9 and an initial learning rate of 0.1. We train each network for 200 epochs with a learning rate scheduler that divides the initial learning rate of 0.1 by a factor of 10 at epoch 80 and epoch 150. All networks are trained with either the log-loss (*i.e.*, cross-entropy) or the Brier score as its loss function. Furthermore, we use a batch size of 128 for CIFAR-10 and CIFAR-100, whereas we use a batch size of 16 for ImageWoof due to the increased size of its images. We also use the preprocessing step mentioned in the paper associated with each network architecture. Typically, this step consists of either per-pixel normalization or per-channel normalization of the images. Furthermore, we train these classifiers either without data augmentation or with "light" data augmentation. We elaborate on "light" data augmentation in Appendix A.4. Unless otherwise stated, no data augmentation is applied and the Brier score is used as the loss function during training.

[2]The ResNet classifiers are trained based on the code from https://github.com/BIGBALLON/cifar-10-cnn.

[3]Excluding LeNet-5, these trained networks have been obtained from https://github.com/markus93/NN_calibration.

A.4 Data augmentation

In this work, we consider two settings for data augmentation, namely (1) no data augmentation and (2) "light" data augmentation. As its name suggests, setting (1) does not apply any form of data augmentation. Setting (2), however, applies random horizontal flips to the images. Additionally, this setting also shifts the image up or down and left or right with a random percentage between 0 and 12.5.

Appendix B

Proofs and derivations

This chapter provides proofs for the corollaries, propositions, and theorems introduced in this work. Furthermore, it provides an estimator for the refinement loss under the Brier score.

B.1 Proof of Theorem 4.3.1

Theorem 4.3.1 (Calibration-refinement decomposition, proven in Appendix B.1)
Let $L^\phi = \mathbb{E}[\phi(\mathbf{Z}, \mathbf{Y})]$ denote the expected loss of confidence score vector $\mathbf{Z}$ with respect to ground truth vector $\mathbf{Y}$ under any proper score rule ϕ. The expected loss L^ϕ can be decomposed as follows:

$$L^\phi = CL^\phi + RL^\phi + DL^\phi, \text{ where } \begin{cases} CL^\phi = \mathbb{E}[d^\phi(\mathbf{Z}, \mathbf{C})] & \text{(Calibration loss)} \\ RL^\phi = \mathbb{E}[d^\phi(\mathbf{C}, \mathbf{Y})] & \text{(Refinement loss)} \\ DL^\phi = \mathbb{E}[\phi(\mathbf{Y}, \mathbf{Y})] & \text{(Dataset loss)} \end{cases}$$

Proof. We omit the proper scoring rule superscript ϕ throughout this proof to improve readability. By definition, proving that $L = CL + RL + DL$ is equivalent to proving that

$$\underbrace{\mathbb{E}_X[\phi(\mathbf{Z}, \mathbf{Y})]}_{L} = \underbrace{\mathbb{E}_X[d(\mathbf{Z}, \mathbf{C})]}_{CL} + \underbrace{\mathbb{E}_X[d(\mathbf{C}, \mathbf{Y})]}_{RL} + \underbrace{\mathbb{E}_X[\phi(\mathbf{Y}, \mathbf{Y})]}_{DL}.$$

Before diving into the proof, we first remark that the expected score divergence $\mathbb{E}_X[d(\mathbf{Z}, \mathbf{Y})]$ can be written as

$$\begin{aligned} \mathbb{E}_X[d(\mathbf{Z}, \mathbf{Y})] &= \mathbb{E}_X[s(\mathbf{Z}, \mathbf{Y}) - s(\mathbf{Y}, \mathbf{Y})] && \text{(Definition 4.2.4)} \\ &= \mathbb{E}_X[\phi(\mathbf{Z}, \mathbf{Y}) - \phi(\mathbf{Y}, \mathbf{Y})] && \text{(Definition 4.2.2)} \\ &= \underbrace{\mathbb{E}_X[\phi(\mathbf{Z}, \mathbf{Y})]}_{L} - \underbrace{\mathbb{E}_X[\phi(\mathbf{Y}, \mathbf{Y})]}_{DL}. \end{aligned}$$

Hence, we can rewrite the last equation as

$$\underbrace{\mathbb{E}_X[\phi(\mathbf{Z},\mathbf{Y})]}_{L} = \mathbb{E}_X[d(\mathbf{Z},\mathbf{Y})] + \underbrace{\mathbb{E}_X[\phi(\mathbf{Y},\mathbf{Y})]}_{DL}.$$

Now, we have to prove that the first term on the RHS is the sum of the CL and RL. Therefore, we have to prove that

$$\mathbb{E}_X[d(\mathbf{Z},\mathbf{Y})] = \underbrace{\mathbb{E}_X[d(\mathbf{Z},\mathbf{C})]}_{CL} + \underbrace{\mathbb{E}_X[d(\mathbf{C},\mathbf{Y})]}_{RL}.$$

Let us start by decomposing $\mathbb{E}_X[d(\mathbf{Z},\mathbf{Y})]$ as follows

$$\begin{aligned}
\mathbb{E}_X[d(\mathbf{Z},\mathbf{Y})] &= \mathbb{E}_X[s(\mathbf{Z},\mathbf{Y}) - s(\mathbf{Y},\mathbf{Y})] && \text{(Definition 4.2.4)}\\
&= \mathbb{E}_X[s(\mathbf{Z},\mathbf{Y}) - s(\mathbf{C},\mathbf{Y}) + s(\mathbf{C},\mathbf{Y}) - s(\mathbf{Y},\mathbf{Y})]\\
&= \mathbb{E}_X[s(\mathbf{Z},\mathbf{Y}) - s(\mathbf{C},\mathbf{Y})] + \mathbb{E}_X[s(\mathbf{C},\mathbf{Y}) - s(\mathbf{Y},\mathbf{Y})]\\
&= \mathbb{E}_X[s(\mathbf{Z},\mathbf{Y}) - s(\mathbf{C},\mathbf{Y})] + \underbrace{\mathbb{E}_X[d(\mathbf{C},\mathbf{Y})]}_{RL}, && \text{(Definition 4.2.4)}
\end{aligned}$$

where the second term on the RHS is equivalent to the refinement loss $RL = \mathbb{E}_X[d(\mathbf{C},\mathbf{Y})]$. All that is left is to show that the first term on the RHS is equivalent to the calibration loss $CL = \mathbb{E}_X[d(\mathbf{Z},\mathbf{C})]$. Using the law of total expectations, we rewrite the first term on the RHS as follows:

$$\begin{aligned}
\mathbb{E}_X[s(\mathbf{Z},\mathbf{Y}) - s(\mathbf{C},\mathbf{Y})] &= \mathbb{E}_{\mathbf{Z}}\Big[\mathbb{E}_X\big[s(\mathbf{Z},\mathbf{Y}) - s(\mathbf{C},\mathbf{Y})|\mathbf{Z}\big]\Big] && \text{(B.1)}\\
&= \mathbb{E}_{\mathbf{Z}}\Big[\mathbb{E}_X\big[s(\mathbf{Z},\mathbf{Y})|\mathbf{Z}\big]\Big] - \mathbb{E}_{\mathbf{Z}}\Big[\mathbb{E}_X\big[s(\mathbf{C},\mathbf{Y})|\mathbf{Z}\big]\Big]. && \text{(B.2)}
\end{aligned}$$

To continue, we show that the two terms in the last equation can be rewritten as

$$\mathbb{E}_{\mathbf{Z}}\Big[\mathbb{E}_X\big[s(\mathbf{Z},\mathbf{Y})|\mathbf{Z}\big]\Big] = \mathbb{E}_{\mathbf{Z}}\big[s(\mathbf{Z},\mathbf{C})\big] \tag{B.3}$$

and

$$\mathbb{E}_{\mathbf{Z}}\Big[\mathbb{E}_X\big[s(\mathbf{C},\mathbf{Y})|\mathbf{Z}\big]\Big] = \mathbb{E}_{\mathbf{Z}}\big[s(\mathbf{C},\mathbf{C})\big]. \tag{B.4}$$

Let us focus on proving that Equation B.3 holds while remarking that Equation B.4 can be derived in a similar fashion. In particular, let us focus on the inner expectation on the LHS of Equation B.3. Since $\mathbf{Y}$ follows from $\mathbf{X}$ and $\mathbf{Z}$

is given, we can also take the expected value over $\mathbf{Y}$ instead of over $\mathbf{X}$. Hence, we can rewrite the LHS of Equation B.3 as follows:

$$\mathbb{E}_{\mathbf{Z}}\Big[\mathbb{E}_X\big[s(\mathbf{Z},\mathbf{Y})|\mathbf{Z}\big]\Big] = \mathbb{E}_{\mathbf{Z}}\Big[\mathbb{E}_{\mathbf{Y}}\big[s(\mathbf{Z},\mathbf{Y})|\mathbf{Z}\big]\Big]. \tag{B.5}$$

Since $\mathbf{Y}$ is a vector with only a 1 at the index of the true class and 0s everywhere else, we remark that $s(\mathbf{p},\mathbf{Y}) = \phi(\mathbf{p},\mathbf{Y})$ for any distribution $\mathbf{p} \in [0,1]^K$ with $\sum_k p_k = 1$. Therefore, we can rewrite the RHS of Equation B.5 as follows:

$$\mathbb{E}_{\mathbf{Z}}\Big[\mathbb{E}_{\mathbf{Y}}\big[s(\mathbf{Z},\mathbf{Y})|\mathbf{Z}\big]\Big] = \mathbb{E}_{\mathbf{Z}}\Big[\mathbb{E}_{\mathbf{Y}}\big[\phi(\mathbf{Z},\mathbf{Y})|\mathbf{Z}\big]\Big]. \tag{B.6}$$

In the last expression, the term $\mathbb{E}_{\mathbf{Y}}[\phi(\mathbf{Z},\mathbf{Y})|\mathbf{Z}]$ represents the average value of $\phi(\mathbf{Z},\mathbf{Y})$ given that the prediction is $\mathbf{Z}$. In this thesis, we defined that $\mathbf{C} = \mathbb{E}_X[\mathbf{Y}|\mathbf{Z}]$ (*i.e.*, C_k gives the probability that the true label of ground truth vector $\mathbf{Y}$ is k). Therefore, the average value of $\phi(\mathbf{Z},\mathbf{Y})$ given that the prediction is $\mathbf{Z}$ can be calculated by letting some variable $\mathbf{y}$ follow distribution $\mathbf{C}$. That is,

$$\begin{aligned}\mathbb{E}_{\mathbf{Y}}[\phi(\mathbf{Z},\mathbf{Y})|\mathbf{Z}] &= \mathbb{E}_{\mathbf{y}\sim\mathbf{C}}[\phi(\mathbf{Z},\mathbf{y})] \\ &= s(\mathbf{Z},\mathbf{C}). && \text{(Definition 4.2.2)}\end{aligned}$$

Using this in the RHS of Equation B.6 gives

$$\mathbb{E}_{\mathbf{Z}}\Big[\mathbb{E}_{\mathbf{Y}}\big[\phi(\mathbf{Z},\mathbf{Y})|\mathbf{Z}\big]\Big] = \mathbb{E}_{\mathbf{Z}}\big[s(\mathbf{Z},\mathbf{C})\big].$$

Hence, we have shown that Equation B.3 holds. In similar fashion, we can show that Equation B.4 holds as well. Using Equation B.3 and Equation B.4, we now show that Equation B.2 equals the calibration loss $CL = \mathbb{E}_X[\phi(\mathbf{Z},\mathbf{C})]$ as follows

$$\begin{aligned}&\mathbb{E}_{\mathbf{Z}}\Big[\mathbb{E}_X\big[s(\mathbf{Z},\mathbf{Y})|\mathbf{Z}\big]\Big] - \\ &\mathbb{E}_{\mathbf{Z}}\Big[\mathbb{E}_X\big[s(\mathbf{C},\mathbf{Y})|\mathbf{Z}\big]\Big] &&= \mathbb{E}_{\mathbf{Z}}[s(\mathbf{Z},\mathbf{C})] - \mathbb{E}_{\mathbf{Z}}[s(\mathbf{C},\mathbf{C})] && \text{(B.7)}\\ &&&= \mathbb{E}_{\mathbf{Z}}[s(\mathbf{Z},\mathbf{C}) - s(\mathbf{C},\mathbf{C})] \\ &&&= \mathbb{E}_{\mathbf{Z}}[d(\mathbf{Z},\mathbf{C})] && \text{(Definition 4.2.4)}\\ &&&= \underbrace{\mathbb{E}_X[d(\mathbf{Z},\mathbf{C})]}_{CL}, && \text{(B.8)}\end{aligned}$$

where Equation B.8 holds since $\mathbf{Z}$ follows from X. Since $\mathbb{E}_X[d(\mathbf{Z},\mathbf{C})]$ equals the LHS of Equation B.7 and Equation B.7 equals Equation B.2, we also have that $\mathbb{E}_X[d(\mathbf{Z},\mathbf{C})]$ equals Equation B.1, which concludes our proof. ■

B.2 Proof of Corollary 4.3.1.1

Corollary 4.3.1.1 (Zero dataset loss, proven in Appendix B.2) For any proper scoring rule ϕ, the calibration-refinement decomposition is given by $L^{\phi} = CL^{\phi} + RL^{\phi} + DL^{\phi}$ (Theorem 4.3.1). For the Brier score ϕ^{BS} and log-loss ϕ^{LL}, we find that $DL^{BS} = 0$ and $DL^{LL} = 0$, respectively. Therefore, their decomposition can be written as $L^{BS} = CL^{BS} + RL^{BS}$ and $L^{LL} = CL^{LL} + RL^{LL}$, respectively.

Proof. Without loss of generality, let $\mathbf{y} \in \{0, 1\}^K$ represent the ground truth vector with ground truth class q, such that $y_q = 1$ and $y_k = 0$ for all $k \neq q$. Let us first consider the Brier score ϕ^{BS}. We find that:

$$\phi^{BS}(\mathbf{y}, \mathbf{y}) = \sum_{k=1}^{K} (y_k - y_k)^2 = 0.$$

Since $\phi^{BS}(\mathbf{y}, \mathbf{y}) = 0$, we find that the dataset loss $DL^{BS} = \mathbb{E}[\phi^{BS}(\mathbf{y}, \mathbf{y})]$ takes a value of 0. Consequently, the calibration-refinement decomposition for the Brier score becomes:

$$\begin{aligned} L^{BS} &= CL^{BS} + RL^{BS} + DL^{BS} \\ &= CL^{BS} + RL^{BS}. \end{aligned}$$

Similarly, for the log-loss ϕ^{LL}, we find that:

$$\phi^{LL}(\mathbf{y}, \mathbf{y}) = -\log y_q = 0.$$

Since $\phi^{LL}(\mathbf{y}, \mathbf{y}) = 0$, we find that the dataset loss $DL^{LL} = \mathbb{E}[\phi^{LL}(\mathbf{y}, \mathbf{y})]$ takes a value of 0. Consequently, the calibration-refinement decomposition for the log-loss becomes:

$$\begin{aligned} L^{LL} &= CL^{LL} + RL^{LL} + DL^{LL} \\ &= CL^{LL} + RL^{LL}. \end{aligned}$$

■

B.3 Refinement loss estimation under the Brier score

This section provides an estimator for the refinement loss under the Brier score. This estimator is given as follows:

$$\widehat{RL}(\mathcal{B}) = \sum_{k=1}^{K} \widehat{RL}_k(\mathcal{B}_k).$$

Before introducing the estimator for the class-wise refinement loss under the Brier score $\widehat{RL}_k(\mathcal{B}_k)$, we first remark that the estimator for the class-wise calibration loss under the Brier score $\widehat{CL}_k$ can be written as:

$$\begin{aligned}\widehat{CL}_k(\mathcal{B}_k) &= \sum_{m=1}^{M} \frac{|B_{k,m}|}{N}(\bar{z}_{k,m} - \bar{c}_{k,m})^2 && \text{(Equation 4.8)} \\ &= \sum_{m=1}^{M} \frac{|B_{k,m}|}{N}\left(\frac{1}{|B_{k,m}|}\sum_{j\in B_{k,m}}\left(z_k^{(j)} - \bar{c}_{k,m}\right)^2\right)\end{aligned}$$

since $\bar{z}_{k,m} = \sum_{j\in B_{k,m}} z_k^{(j)}/|B_{k,m}|$. Using a similar formulation, the class-wise refinement loss under the Brier score $\widehat{RL}_k(\mathcal{B}_k)$ can be estimated as:

$$\widehat{RL}_k(\mathcal{B}_k) = \sum_{m=1}^{M} \frac{|B_{k,m}|}{N}\left(\frac{1}{|B_{k,m}|}\sum_{j\in B_{k,m}}\left(\bar{c}_{k,m} - y_k^{(j)}\right)^2\right).$$

Appendix C

Additional miscalibration and hypotheses results

This chapter provides additional results for experimental evaluations provided throughout this work. Hence, this chapter primarily aims to strengthen claims made throughout this work by providing additional results.

C.1 Miscalibration in neural network classifiers

Throughout this section, we extend the findings in § 3.3 by providing additional results that neural network classifiers often tend to be miscalibrated. Specifically, we show that miscalibration appears for different classifiers and datasets. Moreover, we show that the number of bins for the confidence reliability diagrams does not influence this finding.

C.1.1 Varying the number of bins

In this section, we vary the number of bins on the confidence reliability diagram of ResNet-110 SD, trained with the log-loss and light data augmentation. Figure C.1 illustrates the confidence reliability diagram for 5, 15, 30, and 60 equally-ranged bins. We observe that the number of bins does not influence the observation that the classifier seems miscalibrated concerning confidence calibration.

C.1.2 Miscalibration on CIFAR-10

This section provides additional results that showcase miscalibration in neural network classifiers on CIFAR-10. For this section, we use the same architectures as those used in § 3.3. That is, we use LeNet-5, ResNet-110, ResNet-110 SD, and DenseNet-40, all trained with the log-loss and light data augmentation. We, again, observe that the confidence reliability diagrams for ResNet-110, ResNet-110 SD, and DenseNet-40 show the highest degrees of miscalibration. In contrast, LeNet-5, again, seems relatively well-calibrated. For this dataset, we do, however,

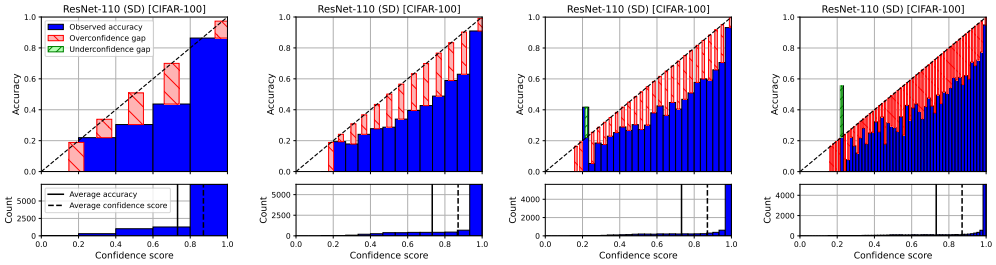


Figure C.1: Illustration of confidence reliability diagrams for ResNet-110 SD on CIFAR-100 for (a) 5, (b) 15, (c) 30, and (d) 60 equally-ranged bins.

note that most confidence scores for the complex networks are contained in the rightmost bin.

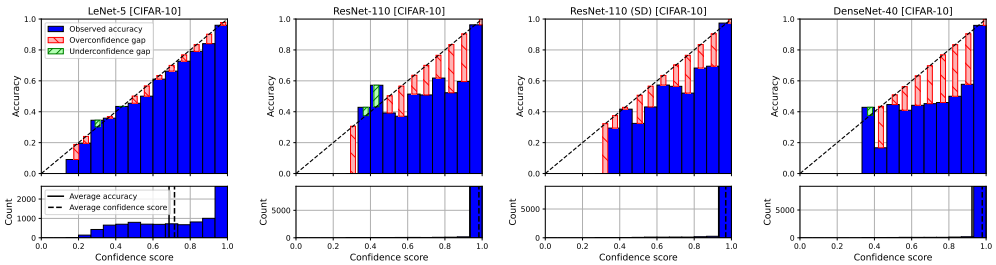
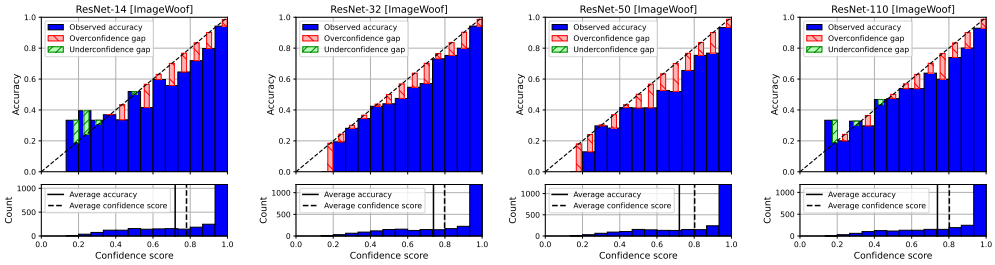


Figure C.2: Illustration of confidence reliability diagrams for (a) LeNet-5, (b) ResNet-110, (c) Resnet-110 SD, and (d) DenseNet-40 on CIFAR-10, all trained with the log-loss and light data augmentation.

C.1.3 Miscalibration on ImageWoof

This section provides additional results that showcase miscalibration in neural network classifiers on ImageWoof. For this section, we use different architectures as those used in § 3.3 and Appendix C.1.2. Specifically, we consider ResNet-14, ResNet-32, ResNet-50, and ResNet-100, all trained with the Brier score and without data augmentation. We change the loss function from the log-loss to the Brier score and we do not use data augmentation to also include different training scenarios. We selected the classifier from the training procedure based on the highest accuracy on the validation set. Figure C.3 illustrates the confidence

reliability diagrams of these classifiers. Again, we observe varying degrees of miscalibration for all classifiers.



(a) **(b)** **(c)** **(d)**

Figure C.3: Illustration of confidence reliability diagrams for (a) ResNet-14, (b) ResNet-32, (c) ResNet-50, and (d) ResNet-110 on ImageWoof, all trained with the Brier score and without data augmentation.

C.2 Positive correlation between the classification error and refinement loss

In this section, we present additional results for the strong positive correlation between the classification error and RL^{BS} in § 4.4.1. Figure C.4 illustrates diagrams of the RL^{BS} and classification error for ResNet-8, ResNet-14, ResNet-32, and ResNet-50 on CIFAR-100, all trained with the Brier score and without data augmentation. Each diagram shows a strong positive correlation between the RL^{BS} and classification, for both the training and test set.

C.3 Comparing the loss function and estimated loss function

In this section, we present additional comparisons of the loss function and the estimated loss function, as discussed earlier in § 4.3.2. Specifically, we focus on the Brier score and the estimated Brier score (*i.e.*, $RL^{BS} + CL^{BS}$) for ResNet-8, ResNet-14, ResNet-32, and ResNet-50 on CIFAR-100 in Figure C.5. We observe that the Brier score and the estimated Brier score align on both the training and test set. These results support our finding in § 4.3.2. In contrast, Figure C.6 shows the log-loss and estimated log-loss (*i.e.*, $RL^{LL} + CL^{LL}$) for ResNet-8, ResNet-14, ResNet-32, and ResNet-50 on CIFAR-100. These figures show that

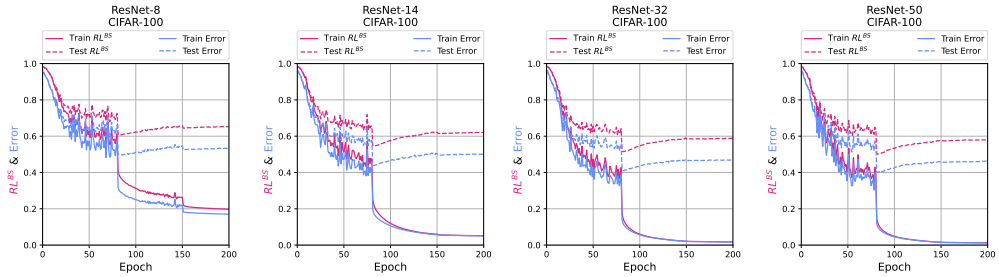


Figure C.4: Illustration of diagrams showcasing the relation between the RL_{BS} and classification error for (a) ResNet-8, (b) ResNet-14, (c) Resnet-32, and (d) ResNet-50 on CIFAR-100, all trained with the Brier score and without data augmentation.

the estimated log-loss does not align with the actual log-loss. These results also support our finding in § 4.3.2.

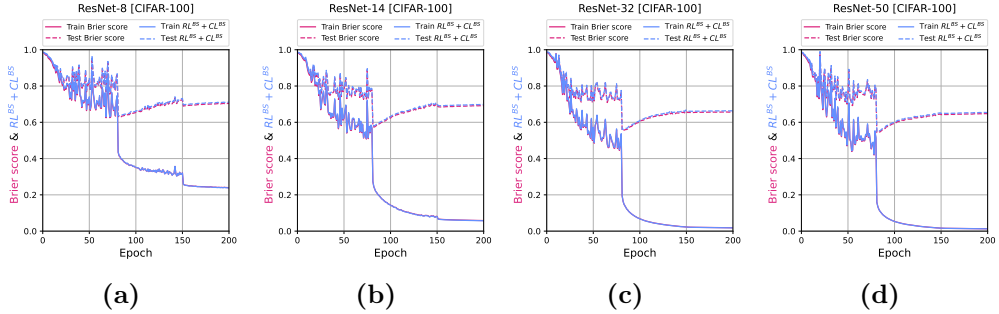
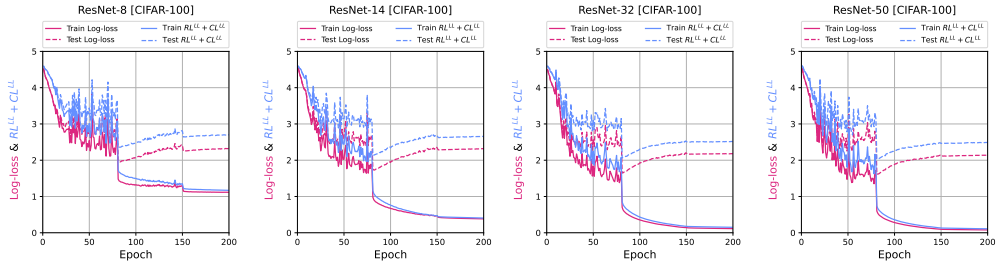


Figure C.5: Illustration of diagrams showcasing the relation between the Brier score and the estimated Brier score, based on $CL^{BS} + RL^{BS}$, for (a) ResNet-8, (b) ResNet-14, (c) ResNet-32, and (d) ResNet-50.

C.4 Additional results for verifying the hypotheses for the Brier score

Throughout this section, we provide additional results that provide support for the overfitting and distribution shift hypotheses, initially tested in § 4.6.4.



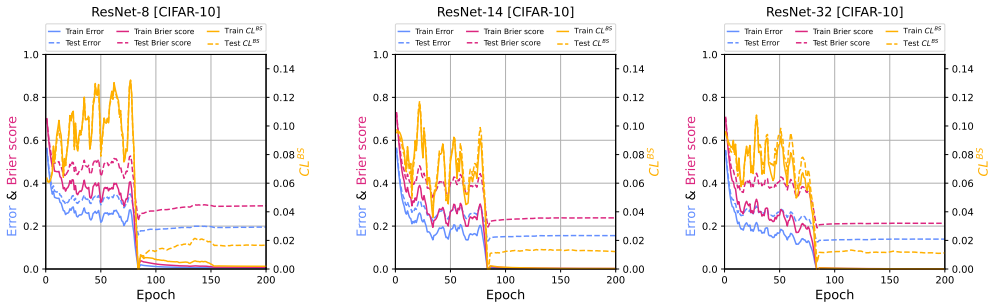
(a) (b) (c) (d)

Figure C.6: Illustration of diagrams showcasing the relation between the log-loss and the estimated log-loss, based on $CL^{LL}+RL^{LL}$, for (a) ResNet-8, ResNet-14, ResNet-32, and ResNet-50.

C.4.1 Hypotheses verification on CIFAR-10

Similar to the diagrams presented in § 4.6.4, Figure C.7 illustrates the classification error, Brier score, and CL^{BS} over the epochs for a collection of training procedures on CIFAR-10. This figure shows that ResNet-14 and ResNet-32 reach a meager Brier score on the training set after epoch 80. Since these classifiers perform exceptionally well on the training set after this epoch, their trained weights will not change much. Since these weights stay relatively similar, all quantities on the training and test set will also stay relatively similar. Consequently, it is challenging to verify the overfitting and distribution shift hypothesis as we start looking at noise. The only classifier for which we still observe a (close to) non-zero loss is ResNet-8. Therefore, we shift our focus to this classifier for the remainder of this section for verifying the hypotheses. For ResNet-8, we can still observe the overfitting and distribution shift hypothesis.

Figure C.8 shows the classification error, Brier score, and conf-ECE_2 over the epochs for ResNet-8. In this figure, the conf-ECE_2 on the training set decreases significantly after epoch 80, whereas the conf-ECE_2 on the test set increases significantly. The relative increase in conf-ECE_2 on the test set is much more substantial than the relative increase in the test error and test Brier score. Hence, we observe the overfitting and distribution shift hypothesis in this figure. In order to obtain a better understanding regarding how miscalibration changes over the epochs, we also include the conf-ECE_2 and CL^{BS} for epochs 81, 115, and 150 in Table C.1. Figure C.9 shows the reliability diagrams associated with these epochs. These diagrams show a substantial difference concerning miscalibration. That is, the classifier from epoch 81 is relatively well-calibrated



(a) **(b)** **(c)**

Figure C.7: Comparison of the classification error, Brier score, and CL^{BS} for (a) ResNet-8, (b) ResNet-14, and (c) ResNet-32 on CIFAR-10. All classifiers have been trained with the Brier score and without data augmentation.

compared to the classifier from epoch 115. Similarly, the classifier from epoch 115 is better calibrated than the classifier from epoch 150.

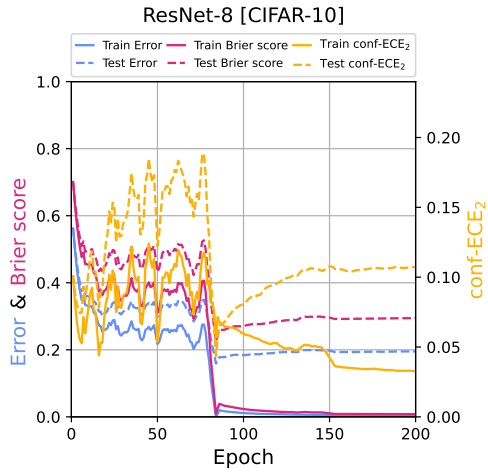


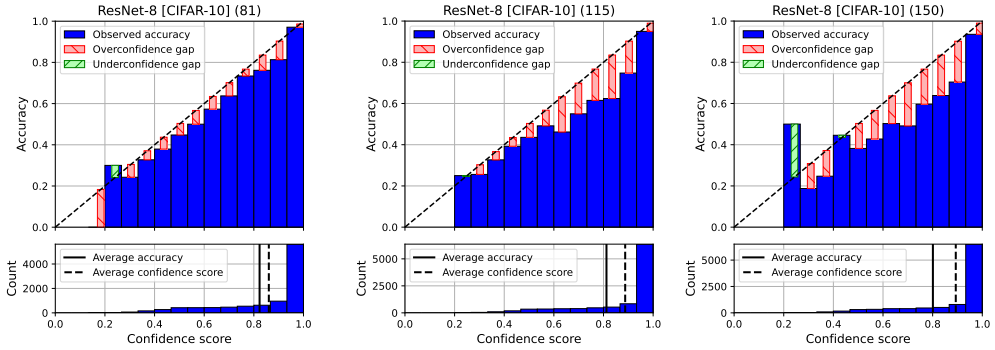
Figure C.8: Comparison of the classification error, Brier score, and conf-ECE$_2$ for ResNet-8 on CIFAR-10.

C.4.2 Hypotheses verification on ImageWoof

We also provide additional results for the experimental evaluation of the classifiers on ImageWoof. Figure C.10 illustrates the calibration error, Brier score,

Table C.1: The conf-ECE$_2$ and CL^{BS} for three different epochs.

Epoch	conf-ECE$_2$	CL^{BS}
81	0.0464	0.0072
115	0.0941	0.0139
150	0.1099	0.0198



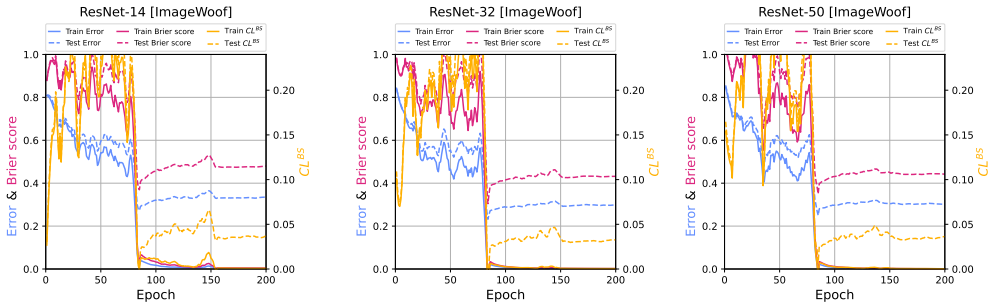
(a) Epoch 81 **(b)** Epoch 115 **(c)** Epoch 150

Figure C.9: Three reliability diagrams of ResNet-8 on CIFAR-10 for (a) epoch 81, (b) epoch 115, and (c) epoch 150.

and CL^{BS} over the epochs for a collection of classifiers on ImageWoof. Similar to in the previous section, we observe that ResNet-32 and ResNet-50 reach a close to zero classification error and Brier score on the training set after epoch 80. However, we can still verify the overfitting and distribution shift hypotheses for all classifiers after epoch 80. We repeat these diagrams for the conf-ECE$_2$ in Figure C.11. For these diagrams, the overfitting and distribution shift hypotheses can also be observed.

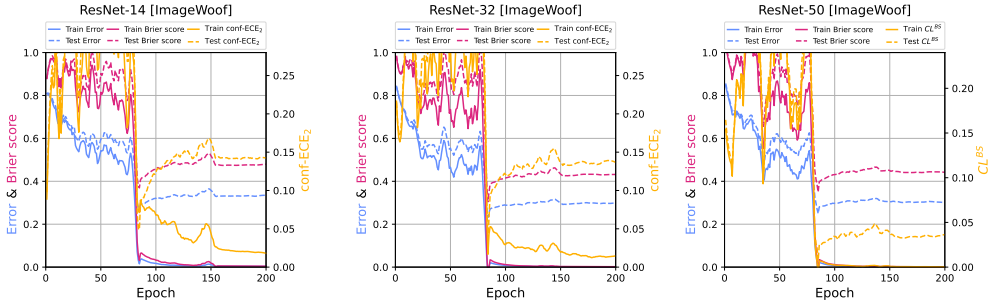
C.5 Hypotheses verification for the log-loss

The experimental evaluation in § 4.6.4 provides support for the overfitting and distribution shift hypotheses by comparing the classification error, Brier score, and calibration loss under the Brier score over the epochs. The classifiers for this experimental evaluation were trained with the Brier score. In this section, we want to find support for these hypotheses for classifiers trained with the log-loss, as the log-loss is another popular loss function. In § 4.4.1, we found that we



(a) **(b)** **(c)**

Figure C.10: Comparison of the classification error, Brier score, and CL^{BS} for (a) ResNet-14, (b) ResNet-32, and (c) ResNet-50 on ImageWoof. All classifiers have been trained with the Brier score and without data augmentation.



(a) **(b)** **(c)**

Figure C.11: Comparison of the classification error, Brier score, and conf-ECE_2 for (a) ResNet-14, (b) ResNet-32, and (c) ResNet-50 on ImageWoof. All classifiers have been trained with the Brier score and without data augmentation.

cannot accurately estimate the calibration loss under the log-loss. Therefore, we continued our experimental evaluation with the Brier score. Hence, we believe that we should not use the calibration loss under the log-loss to verify the hypotheses for classifiers trained with the log-loss. Although not as clean, comparing the classification error and log-loss with other calibration metrics, such as the conf-ECE_2 and CW-ECE_2, still provides insights into whether miscalibration stems from the causes proposed in the hypotheses.

To that end, we repeat the experimental evaluation in § 4.6.4 for the same classifiers trained with the log-loss while focusing on the conf-ECE_2 and CW-ECE_2 instead of the CL^{BS}. Figure C.12 illustrates the classification error, log-

loss (scaled by a factor of 4), and conf-ECE$_2$ over the epochs. In this figure, we observe both the overfitting and distribution shift hypothesis for all classifiers. For ResNet-32, ResNet-50, and ResNet-110, we observe the overfitting to a lesser degree than for ResNet-14. We believe this observation stems from ResNet-32, ResNet-50, and ResNet-110 already reaching a close-to-zero loss right after epoch 80. Figure C.13 illustrates the classification error, log-loss (scaled by a factor of 4), and CW-ECE$_2$ over the epochs. In this figure, we can also observe both hypotheses for all classifiers, although it is significantly more challenging to find support for the hypotheses in the diagrams of ResNet-32, ResNet-50, and ResNet-110. Again, we believe this to stem from these classifiers already reaching a close-to-zero loss right after epoch 80. Regardless, we find support for the overfitting and distribution shift hypotheses for classifiers trained with the log-loss based on the conf-ECE$_2$ and CW-ECE$_2$.

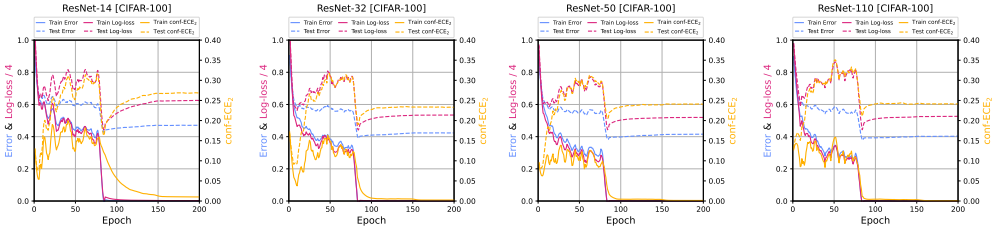


Figure C.12: Comparison of the classification error, log-loss, and conf-ECE$_2$ for (a) ResNet-14, (b) ResNet-32, (c) ResNet-50, and (d) ResNet-110 on CIFAR-100, all trained with the log-loss without data augmentation.

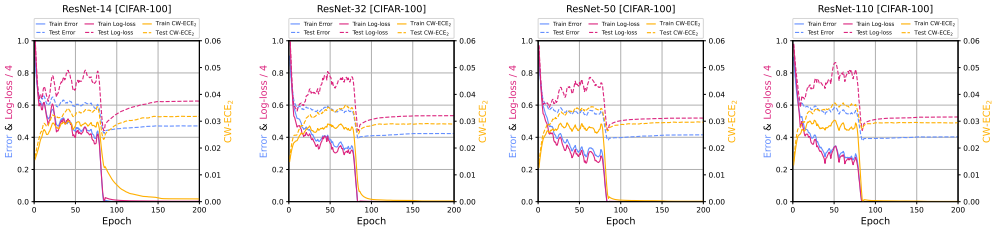


Figure C.13: Comparison of the classification error, log-loss, and CW-ECE$_2$ for (a) ResNet-14, (b) ResNet-32, (c) ResNet-50, and (d) ResNet-110 on CIFAR-100, all trained with the log-loss without data augmentation.

Appendix D

Additional calibration method results

This chapter provides additional results for the evaluation of calibration methods, initially presented in § 7.2. In contrast to § 7.2, this chapter evaluates the calibration methods for classifiers trained *without* data augmentation. Similar to § 7.2, we consider ResNet-8, ResNet-14, ResNet-32, ResNet-50, ResNet-110, and ResNet-152 on CIFAR-10, CIFAR-100, and ImageWoof for this evaluation.

D.1 Confidence-ECE

The conf-ECE_2 for each classifier, dataset, and calibration method is depicted in Table D.1. Each table cell for the uncalibrated classifier contains the conf-ECE_2 value together with the accuracy of the classifier as a percentage. The table cells of all calibration methods also contain the conf-ECE_2 value but together with the percentage point increase or decrease in accuracy compared to the uncalibrated classifier.

In this table, we observe similar trends as in § 7.2.1. That is, temperature scaling performs well across all classifiers and datasets. It also shows that vector scaling works well but might impact the accuracy, either positively or negatively. Furthermore, matrix scaling also seems promising on CIFAR-10; however, it leads to a massive drop in accuracy on CIFAR-100. We also observe that data augmentation leads to a substantial increase in accuracy. Hence, a practitioner would generally select the classifiers trained with data augmentation instead of those without. This is the reason why we limit ourselves to the classifiers trained with data augmentation in § 7.2.1. We observe that data augmentation always leads to a better conf-ECE_2 for the classifiers on CIFAR-10. In contrast, the conf-ECE_2 tends to become worse with data augmentation compared to the uncalibrated classifier on CIFAR-100 and ImageWoof. This observation can be explained by the classifiers changing substantially, which is indicated by their substantial increase in accuracy. Remark 3.3.1 argued that a classifier with an

extremely high accuracy intuitively has an easier time achieving good calibration compared to one with lower accuracy. Our observations do not contradict this remark since the classifiers for CIFAR-100 and ImageWoof do not have extremely high accuracy. Generally, we observe that all findings done in § 7.2.1 can also be observed for the classifiers trained without data augmentation.

Table D.1: The conf-ECE_2 for multiple classifiers on several datasets. The values in the cells indicate the conf-ECE_2 together with the percentage point increase or decrease in accuracy compared to the uncalibrated classifier in parenthesis. The "uncalibrated" column contains the accuracy of the base classifier instead. These classifiers are trained with the Brier score, *without* data augmentation. **Bold** values indicate the best performing techniques concerning the conf-ECE_2 and underlined values indicate the best performing techniques concerning the accuracy.

Dataset	Classifier	Mitigative								Preventative	
		Accuracy preserving		Non-accuracy preserving							
		Uncalibrated	TS	PS	IR	HB	BC	VS	MS	DA	CSS
CIFAR-10	ResNet-8	0.061 (82.3)	**0.013 (0.0)**	0.161 (-0.24)	0.019 (-0.02)	0.039 (-0.25)	0.019 (0.22)	0.017 (0.39)	0.017 (-0.07)	0.031 (4.1)	0.045 (-0.1)
CIFAR-10	ResNet-14	0.081 (85.0)	0.019 (0.0)	0.12 (-0.61)	0.026 (-0.08)	0.044 (-0.22)	0.023 (-0.12)	0.018 (-0.1)	**0.017 (-0.15)**	0.061 (4.7)	0.062 (-0.1)
CIFAR-10	ResNet-32	0.08 (86.5)	**0.021 (0.0)**	0.087 (-0.51)	0.033 (-0.1)	0.061 (-0.03)	0.026 (-0.04)	0.027 (-0.04)	0.026 (-0.32)	0.062 (5.5)	0.08 (0.0)
CIFAR-10	ResNet-50	0.079 (87.7)	**0.019 (0.0)**	0.066 (-0.14)	0.03 (-0.11)	0.057 (-0.26)	0.022 (-0.04)	0.02 (0.07)	0.022 (-0.04)	0.062 (4.6)	0.070 (-0.15)
CIFAR-10	ResNet-110	0.08 (87.6)	0.029 (0.0)	0.061 (-0.11)	0.028 (-0.11)	0.058 (-0.4)	0.026 (-0.08)	0.02 (-0.07)	**0.018 (-0.16)**	0.062 (5.3)	0.08 (-0.16)
CIFAR-10	ResNet-152	0.078 (87.6)	**0.021 (0.0)**	0.056 (0.05)	0.027 (0.07)	0.052 (0.01)	0.023 (0.12)	0.023 (0.21)	0.022 (0.09)	0.061 (5.7)	0.076 (-0.03)
CIFAR-100	ResNet-8	0.039 (51.0)	0.022 (0.0)	0.258 (-2.09)	0.033 (-0.68)	0.088 (-3.42)	0.013 (0.09)	**0.012 (0.12)**	0.23 (-10.01)	0.056 (3.8)	0.039 (0.0)
CIFAR-100	ResNet-14	0.095 (55.2)	0.02 (0.0)	0.279 (-1.09)	0.044 (-0.29)	0.098 (-2.82)	0.026 (0.21)	**0.015 (0.26)**	0.209 (-8.21)	0.101 (7.1)	0.095 (0.0)
CIFAR-100	ResNet-32	0.115 (58.6)	**0.023 (0.0)**	0.269 (-1.15)	0.061 (-0.5)	0.11 (-2.93)	0.043 (0.07)	0.032 (-0.03)	0.211 (-8.27)	0.141 (6.9)	0.115 (0.0)
CIFAR-100	ResNet-50	0.128 (59.3)	**0.029 (0.0)**	0.26 (-1.24)	0.065 (-0.32)	0.106 (-2.75)	0.05 (0.08)	0.031 (-0.05)	0.194 (-7.49)	0.155 (7.9)	0.115 (-0.2)
CIFAR-100	ResNet-110	0.137 (59.9)	**0.026 (0.0)**	0.257 (-0.72)	0.064 (-0.5)	0.105 (-2.84)	0.048 (0.0)	0.034 (-0.38)	0.2 (-7.04)	0.165 (8.9)	0.124 (-0.11)
CIFAR-100	ResNet-152	0.133 (59.8)	**0.032 (0.0)**	0.251 (-0.7)	0.062 (-0.31)	0.105 (-2.89)	0.05 (0.18)	**0.032 (0.04)**	0.193 (-7.54)	0.162 (9.6)	0.137 (-0.01)
ImageWoof	ResNet-8	0.078 (63.9)	0.039 (0.0)	0.213 (0.267)	0.045 (1.031)	0.059 (0.076)	0.036 (1.489)	0.034 (1.45)	0.028 (1.183)	**0.025 (9.2)**	0.078 (0.0)
ImageWoof	ResNet-14	0.083 (71.9)	0.042 (0.0)	0.198 (-1.527)	0.054 (-0.954)	0.08 (-1.489)	0.037 (-0.153)	0.037 (-0.534)	**0.035 (-1.374)**	0.056 (7.8)	0.065 (-1.52)
ImageWoof	ResNet-32	0.067 (73.9)	0.035 (0.0)	0.195 (-1.069)	0.041 (-0.611)	0.066 (-1.183)	0.034 (-0.305)	0.034 (-0.229)	**0.022 (-1.145)**	0.088 (7.7)	0.067 (0.0)
ImageWoof	ResNet-50	0.093 (72.0)	0.037 (0.0)	0.192 (-0.382)	0.054 (0.382)	0.069 (-0.267)	0.038 (0.42)	0.031 (0.382)	**0.03 (0.229)**	0.093 (10.0)	0.086 (-0.1)
ImageWoof	ResNet-110	0.076 (73.9)	0.043 (0.0)	0.192 (-0.305)	0.038 (-0.038)	0.053 (-0.84)	0.044 (0.42)	**0.042 (0.229)**	0.043 (0.42)	0.095 (8.4)	0.076 (0.0)
ImageWoof	ResNet-152	0.028 (76.0)	0.031 (0.0)	0.189 (-0.534)	0.038 (0.229)	0.048 (-0.267)	0.025 (0.0)	0.029 (0.382)	**0.023 (-0.267)**	0.075 (5.9)	0.026 (-0.2)

D.2 Class-wise ECE

The CW-ECE_2 for each classifier, dataset, and calibration method is depicted in Table D.2. Each table cell for the uncalibrated classifier contains the CW-ECE_2 value together with the accuracy of the classifier as a percentage. The table cells of all calibration methods also contain the CW-ECE_2 value but together with the percentage point increase or decrease in accuracy compared to the uncalibrated classifier.

Table D.2: The CW-ECE$_2$ for multiple classifiers on several datasets. The values in the cells indicate the CW-ECE$_2 \cdot 10^2$ together with the percentage point increase or decrease in accuracy compared to the uncalibrated classifier in parenthesis. The "uncalibrated" column contains the accuracy of the base classifier instead. These classifiers are trained with the Brier score, *without* data augmentation. **Bold** values indicate the best performing techniques concerning the CW-ECE$_2$ and underlined values indicate the best performing techniques concerning the accuracy.

Dataset	Classifier	Mitigative								Preventative	
		Accuracy preserving		Non-accuracy preserving							
		Uncalibrated	TS	PS	IR	HB	BC	VS	MS	DA	CSS
CIFAR-10	ResNet-8	2.702 (82.3)	1.97 (0.0)	5.899 (-0.24)	1.883 (-0.02)	2.362 (-0.25)	1.927 (0.22)	**1.43 (0.39)**	1.698 (-0.07)	2.262 (4.1)	2.209 (-0.1)
CIFAR-10	ResNet-14	3.175 (85.0)	1.931 (0.0)	4.806 (-0.61)	2.163 (-0.08)	2.52 (-0.22)	2.04 (-0.12)	1.774 (-0.1)	**1.706 (-0.15)**	2.65 (4.7)	2.588 (-0.1)
CIFAR-10	ResNet-32	3.211 (86.5)	1.931 (0.0)	3.712 (-0.51)	2.035 (-0.1)	2.673 (-0.03)	2.015 (-0.04)	1.987 (-0.04)	**1.888 (-0.32)**	2.564 (5.5)	3.211 (0.0)
CIFAR-10	ResNet-50	3.185 (87.7)	1.912 (0.0)	3.082 (-0.14)	1.962 (-0.11)	2.558 (-0.26)	1.924 (-0.04)	1.762 (0.07)	**1.601 (-0.04)**	2.691 (4.6)	2.779 (-0.15)
CIFAR-10	ResNet-110	3.244 (87.6)	2.097 (0.0)	2.858 (-0.11)	2.026 (-0.11)	2.671 (-0.4)	1.961 (-0.08)	1.757 (-0.07)	**1.653 (-0.16)**	2.917 (5.3)	3.159 (-0.16)
CIFAR-10	ResNet-152	3.157 (87.6)	2.039 (0.0)	2.812 (0.05)	2.098 (0.07)	2.492 (0.01)	1.871 (0.12)	**1.734 (0.21)**	1.848 (0.09)	2.798 (5.7)	3.017 (-0.03)
CIFAR-100	ResNet-8	1.7 (51.0)	1.659 (0.0)	2.802 (-2.09)	1.639 (-0.68)	1.751 (-3.42)	**1.61 (0.09)**	1.617 (0.12)	2.821 (-10.01)	1.878 (3.8)	1.7 (0.0)
CIFAR-100	ResNet-14	1.806 (55.2)	1.62 (0.0)	2.927 (-1.09)	1.653 (-0.29)	1.769 (-2.82)	1.639 (0.21)	**1.614 (0.26)**	2.661 (-8.21)	2.016 (7.1)	1.806 (0.0)
CIFAR-100	ResNet-32	2.037 (58.6)	1.711 (0.0)	2.854 (-1.15)	1.763 (-0.5)	1.951 (-2.93)	1.738 (0.07)	**1.681 (-0.03)**	2.676 (-8.27)	2.157 (6.9)	2.037 (0.0)
CIFAR-100	ResNet-50	2.117 (59.3)	1.726 (0.0)	2.806 (-1.24)	1.79 (-0.32)	1.886 (-2.75)	1.78 (0.08)	**1.718 (-0.05)**	2.614 (-7.49)	2.311 (7.9)	1.997 (-0.2)
CIFAR-100	ResNet-110	2.177 (59.9)	1.697 (0.0)	2.705 (-0.72)	1.812 (-0.5)	1.857 (-2.84)	1.722 (0.0)	**1.659 (-0.38)**	2.573 (-7.04)	2.338 (8.9)	2.117 (-0.11)
CIFAR-100	ResNet-152	2.139 (59.8)	1.732 (0.0)	2.701 (-0.7)	1.835 (-0.31)	1.829 (-2.89)	1.714 (0.18)	**1.686 (0.04)**	2.525 (-7.54)	2.287 (9.6)	2.13 (-0.01)
ImageWoof	ResNet-8	4.892 (63.9)	4.813 (0.0)	7.737 (0.267)	4.229 (1.031)	4.087 (0.076)	4.135 (1.489)	3.574 (1.45)	**3.372 (1.183)**	3.214 (9.2)	4.892 (0.0)
ImageWoof	ResNet-14	4.465 (71.9)	3.203 (0.0)	7.127 (-1.527)	3.985 (-0.954)	4.51 (-1.489)	3.13 (-0.153)	3.479 (-0.534)	**3.031 (-1.374)**	3.901 (7.8)	3.892 (-1.52)
ImageWoof	ResNet-32	3.787 (73.9)	3.456 (0.0)	7.27 (-1.069)	3.582 (-0.611)	4.202 (-1.183)	3.283 (-0.305)	3.377 (-0.229)	**3.101 (-1.145)**	4.316 (7.7)	3.787 (0.0)
ImageWoof	ResNet-50	4.311 (72.0)	3.683 (0.0)	7.087 (-0.382)	3.884 (0.382)	3.97 (-0.267)	3.566 (0.42)	**3.169 (0.382)**	3.513 (0.229)	4.604 (10.0)	4.311 (-0.1)
ImageWoof	ResNet-110	4.611 (73.9)	4.138 (0.0)	6.83 (-0.305)	3.47 (-0.038)	3.406 (-0.84)	3.795 (0.42)	3.728 (0.229)	**3.24 (0.42)**	4.278 (8.4)	4.611 (0.0)
ImageWoof	ResNet-152	3.846 (76.0)	3.648 (0.0)	6.984 (-0.534)	3.946 (0.229)	3.916 (-0.267)	3.708 (0.0)	3.335 (0.382)	**3.085 (-0.267)**	4.076 (5.9)	3.820 (-0.2)

Similar to the previous section, we also observe similar trends in this table as in § 7.2.2. Notably, matrix scaling and vector scaling outperform the other calibration methods across most classifiers and datasets. Again, matrix scaling substantially decreases accuracy on CIFAR-100, which renders it less favorable in a real-world use case. We also observe that data augmentation sometimes leads to better-calibrated classifiers. In the cases that data augmentation results in worse calibration, we suspect the primary cause of this to be the substantial improvement in accuracy. Due to this substantial improvement in accuracy, it becomes more challenging to compare the results for data augmentation with the results for the uncalibrated classifier. In the future, it might be interesting to evaluate two classifiers trained with different degrees of data augmentation that result in roughly similar accuracies. Then, it would be interesting if higher degrees of data augmentation help reduce the degree of miscalibration, despite it not impacting the accuracy.