Czech Technical University
Faculty of Electrical Engineering
Department of Cybernetics

Master's thesis

Rectifying Probabilistic Predictions of Neural Networks

Bc. Tuan Anh Ho

Supervisor: Mgr. Oleksandr Shekhovtsov, Ph.D.

Study Programme: Open Informatics
Field of Study: Data Science
January 2023

MASTER‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name:	**Ho Tuan Anh**	Personal ID number:	**474375**
Faculty / Institute:	**Faculty of Electrical Engineering**		
Department / Institute:	**Department of Computer Science**		
Study program:	**Open Informatics**		
Specialisation:	**Data Science**		

II. Master’s thesis details

Master’s thesis title in English:

Rectifying Probabilistic Predictions of Neural Networks

Master’s thesis title in Czech:

Opravování pravděpodobnostních predikcích neuronových sítí

Guidelines:

Neural networks are widely applied in recognition problems. Commonly, they are trained on a typically very large training set by optimizing a probabilistic learning criterion, e.g. likelihood. The architecture and the learning scheme are tuned by researchers towards good generalization with respect to recognition accuracy. When the output of the trained classifier includes predictive probability, which is needed in a number of tasks, accurate values of predictive probabilities become necessary. However, the probabilistic predictions are typically not reliable and even misleading, e.g. overconfident w.r.t. the real error rate. It is thus required to correct the model when knowing the test task and test conditions and using a small (labeled) realistic data set. In the thesis:
1. Analyze the issue of misleading probabilistic predictions: in which scenarios it may be detrimental (change of the loss matrix, prior shift, domain shift, out-of-domain data)? We can possibly adjust the model or predictions at different stages: training, calibration, test. Discuss these options. Relate to the forecasting literature [5,6].
2. Assume that the network will be used for statistical decision making with a known loss matrix. Based on the results of the semestral project, experimentally compare, analyze and develop calibration methods for the known loss matrix and labeled data.
3. Assume that at the test time the distribution of classes is different from the training and is known or can be estimated by existing methods. If the model was calibrated, the Bayesian correction of the predictive distribution is expected to perform well. Calibrate the predictive distribution for this purpose by extending the techniques from 2).
4. Propose and test approaches in the following directions:
a. Improve calibration and correct for the prior shift (part 2) when the test class distribution is not known but we have an unlabeled data sample.
b. If the model is well-calibrated but there is a domain shift at test time (or even out-of-domain examples are presented to it) the probabilistic predictions can become misleading again. Can the domain shift adaptation and calibration be addressed in a unified framework?
5. Test the methods on realistic datasets (e.g., mushrooms [7], snakes, skin cancer [2], semantic segmentation of road scenes).

Bibliography / sources:

[1] Sipka T., Sulc M and Matas J. (2022) The Hitchhiker’s Guide to Prior-Shift Adaptation.
[2] Zhao S., et al. (2021) Calibrating predictions to decisions: A novel approach to multi-class calibration.
[3] Pampari A…. Ermon S. (2020) Unsupervised Calibration under Covariate Shift
[4] Sahoo R… Ermon S. (2021) Reliable Decisions with Threshold Calibration
[5] DeGroot and Fienberg (1982). Assessing probability assessors: calibration and refinement
[6] Murphy and Winkler (1987). A general framework for forecast verification.
[7] Picek, L. et al..(2021) Danish fungi 2020-not just another image recognition dataset

Name and workplace of master’s thesis supervisor:

Mgr. Oleksandr Shekhovtsov, Ph.D. Visual Recognition Group FEE

Name and workplace of second master’s thesis supervisor or consultant:

Date of master’s thesis assignment: **15.09.2022** Deadline for master's thesis submission: **10.01.2023**

Assignment valid until: **19.02.2024**

Mgr. Oleksandr Shekhovtsov, Ph.D.	Head of department’s signature	prof. Mgr. Petr Páta, Ph.D.
Supervisor’s signature		Dean’s signature

III. Assignment receipt

The student acknowledges that the master’s thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the master’s thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student’s signature

Acknowledgement

Firstly, I would like to thank FEE CTU in general for not only preparing me more than well for the life ahead of me, but also for giving me friends, maybe future colleagues and experiences that I will never forget.

Next, I would like to thank prof. Jiří Matas for allowing me to be surrounded by the incredible people in the VRG and to work on the project. Discussions of his or our ideas were always motivational. This also had an immense impact on my career path as I was able to get in touch with like-minded people and also gain experience with research.

A thank you also belongs to Milan Šulc, Ph.D. for starting this project and taking me under his wing in the beginnings of the project before he left.

Nevertheless, I would like to express my most sincere gratitude to the supervisor of this thesis, Oleksandr Shekhovtsov, Ph.D. for not only leading this project and guiding me through it, but also for the invaluable lessons he gave me ranging from math and political discussions to opinions on how one should raise children. Besides, even though he had every right to treat me just as his student, I felt like he treated me like an equal colleague, which made it that much more easy to work with him, as I was never afraid of asking questions. I have sincerely enjoyed spending my time with him either by talking about the project or just by being ourselves and for that I am grateful.

Further, a big thank you belongs to my family – sister, mom, brother–in–law, and friends – my flatmates with whom I lived for the last five or six years and they were there for me through the good and bad, and of course, the friends I made here at CTU belong here too. Yes, this includes you, Klára, as well. Does this sound familiar?

Finally, as I sort of end my academic career with this thesis (for now at least), I hope that my late father would not only be proud of me, but more importantly proud of himself. His children will live a much more comfortable life thanks to his and my mom's hard work (the two people that literally sacrificed everything they had at a certain point in their lives for their children and relatives overseas). Rest well.

Author’s statement

Abstract

The use of neural networks in classification tasks is very common. They are frequently taught on a training set that is typically quite large by optimizing a probabilistic learning criterion, such as likelihood. Researchers have optimized the architecture and the learning strategy for good generalization in terms of classification accuracy. Accurate probability predictions are required when the probabilistic output of the trained classifier is used in a downstream decision problem. The probabilistic predictions, however, are frequently unreliable and even deceptive, for example, overconfidence of the classifier relative to the actual error rate. Utilizing a small dataset while having the knowledge of the test task and test conditions, we are capable of updating the model for an improved performance. In the work, we give an overview of such model updates and propose new loss functions, namely Direct Loss and Integral Loss that allow for improvements in a more specialized manner based on the test task. We show the performance of these loss functions in combinations with certain calibration parametrizations on number of realistic experiments, which include non 0/1 loss matrices and prior shift adaptation.

Keywords Probability Calibration, Neural Networks, Classification, Decision-making, Confident Predictions

Abstrakt

Použití neuronových sítí v klasifikačních úlohách je velmi běžné. Často se vyučují na tréninkové sadě, která je obvykle poměrně velká, optimalizací pravděpodobnostního učebního kritéria, jako je věrohodnost. Výzkumníci optimalizovali architekturu a strategii učení pro dobrou generalizaci z hlediska přesnosti klasifikace. Přesné predikce pravděpodobnosti jsou vyžadovány, když je pravděpodobnostní výstup trénovaného klasifikátoru použit v následném rozhodování. Pravděpodobnostní predikce jsou však často nespolehlivé a dokonce klamavé, například přílišně velké sebevědomí klasifikátoru vzhledem ke skutečné chybovosti. S využitím malé datové sady a znalosti testovací úlohy a testovacích podmínek jsme tedy schopni aktualizovat model pro lepší výkon. V této práci dáváme přehled takových aktualizací modelů a navrhujeme nové ztrátové funkce, jmenovitě Direct Loss a Integral Loss, které umožňují vylepšení specializovanějším způsobem na základě požadavků během testování. Ukážeme výkon těchto ztrátových funkcí v kombinaci s určitými kalibračními parametrizace na řadě realistických experimentů, které zahrnují non 0/1 ztrátové matice a změnu apriorních pravědpodobností.

Klíčová slova Kalibrace pravděpodobností, Neuronové sítě, Klasifikace, Rozhodování, Věrohodné predikce

Contents

List of Figures

Acronyms

BCTS Bias-Corrected Temperature Scaling. 17, 18, 20, 26, 47–52, 57, 59

CE Expected Calibration Error. 13, 20

CV Cross Validation. 23

ECE Empirical Expected Calibration Error. 13–16, 18, 20, 50, 52, 53, 55

ERM Empirical Risk Minimization. 9, 20

KPI Key Performance Indicator. 3, 5

MMCE Maximum Mean Calibration Error. 16, 21

MS Matrix Scaling. 20

NLL Negative log-likelihood. 10, 11, 16–20, 25, 46, 47, 49–59

SCE Static Calibration Error. 14, 15

TS Temperature Scaling. 17, 19, 26, 46–52, 59

VS Vector Scaling. 18, 20, 26, 47–50, 52, 57, 59

Chapter 1

Introduction

1.1 Motivation

The reliability of probabilistic forecasters has its roots mainly in meteorology. Consider the following example: One would anticipate that, out of all days with a 25% probability of rain, around a quarter would be rainy and the rest to differ. The predictor is considered to be calibrated if all predictions show this to be the case. Even while we would prefer for the forecaster to be accurate, if that is not possible, we would nevertheless wish for it to be calibrated – that is, to properly reflect what it does not know. Wanting a good predictive model to make accurate predictions in proportion to its level of confidence is clearly justifiable, however, many models do not have this fundamental property (Niculescu-Mizil & Caruana 2005; Platt et al. 1999). Many downstream layers / decision-makers that build upon these miscalibrated models will, as a result, make inaccurate decisions. In safety-critical domains like medical diagnosis, autonomous driving etc. the effects of an overconfidently wrong decision can be devastating, implying that we are in need to calibrated predictive models.

Calibration in Deep learning

As deep learning gains on popularity, especially in fields like computer vision as they dominate the field regarding their performance, naturally we are to question whether the deep learning classifiers are calibrated. Even the simple most confident prediction calibration, that is making the prediction with the highest probability reflect the knowledge of the model, is currently not easy to achieve, as we have not designed a unified and general measure of calibration (Nixon et al., 2019). We seem to come to a disagreement on whether neural networks are calibrated or not.

Firstly, we are told that neural networks are reasonably calibrated in (Niculescu-Mizil & Caruana, 2005), followed by (Guo et al., 2017), where the authors say that many deep neural networks are not calibrated and then in (Minderer et al., 2021) we are told that they are. Tackled by (Carrell et al., 2022), it is said that there seems to be a general disagreement in the approach how calibration is measured as deep neural networks are still a large category containing convolutional neural networks, transformers, etc..

Refined notions of calibration

First calibration notions were discussed in (Murphy & Winkler 1987; DeGroot & Fienberg 1981; Bröcker 2009), performance functions were presented as a decomposition that included both uncertainty and reliability – i.e. how reliable one was during the evaluation of the performance. With the possible downfalls of reliability, calibration theory developed into the many notions that we have. While many applications are concerned with calibrating only the most confident prediction (Zadrozny & Elkan 2002; Guo et al. 2017), in many tasks we require all the predictions to be calibrated – distribution calibration as mentioned in (Vaicenavicius et al. 2019). Discussed in (Widmann et al. 2019; Zhao et al. 2021), achieving this type of calibration is a eminently difficult, demanding an unreasonably large amount of data proportionally to the number of classes.

Calibration Methods

Calibration is done mainly using some kind of parametrization or by transforming the predicted probabilities into the calibrated probabilities. The latter's most known approach is to fit a function using isotonic regression (Niculescu-Mizil & Caruana, 2005). Parametrization involves learning some parameters that are somehow combined with the output logits with the goal that the resulting probability will be calibrated. Here, the most famous parametrization is Temperature Scaling (Guo et al., 2017), which learns a single parameter $T > 0$ that subsequently divides the logits of the neural network. Some other parametrizations include Platt Scaling or Bias-Corrected Temperature Scaling (Platt et al. 1999; Alexandari et al. 2020). Calibration has also been approached in different stages of the training. The most widely used approach is the so-called post-hoc calibration, where the calibration is done for a given blackbox probabilistic predictor (Zadrozny & Elkan 2002; Guo et al. 2017; Zhao et al. 2021; Sahoo et al. 2021; Platt et al. 1999). Further, there have been experiments to calibrate during the training stage, i.e. train a better calibrated predictor Kumar et al. (2018).

Role of Downstream Tasks

Assuming there would be a method to help us achieve calibrated predictions, besides having reliable outputs, we would also be capable of solving other ensuing tasks, e.g. decision-making under non-0/1 loss functions (Zhao et al., 2021) or predictions under distributions different from the training data – dataset shifts. These have been tackled by e.g. using the (uncalibrated) predictions to estimate the dataset shift (Alexandari et al. 2020; Sipka et al. 2021; Pampari & Ermon 2020) or by simulating many dataset shifts and calibrating with respect to the augmented dataset (Tomani et al., 2021). To the same end, one might want to calibrate the probabilistic outputs so that the main goal will be to improve the performance under a dataset shift or making a decision under a different loss instead of reflecting the true reliability.

1.2 Different problem setups

Let us first discuss several circumstances where the predictive probabilities rather than merely classification decisions are significant before concentrating on the specific issues that the thesis addresses. Additionally, we are not interested in calibration for the sake of calibration in this work; rather, we are interested in calibration to enhance performance in downstream tasks. In order to examine practical use cases where probabilistic predictions might be improved or rectified, we want to investigate when they can be deceptive for certain activities. We restrict the discussion to models trained for classification.

1.2.1 How Misleading Probabilistic Predictions (Lack of Calibration) Can Impact Downstream Tasks

- A decision is made after interpreting the probability of the predicted class as confidence (e.g., class "apple," 80% confidence) (e.g. decide whether the result is trustworthy for further processing or should be skipped or more observations should be acquired). Confidence is vital for safety-sensitive applications.
- The classification accuracy as a Key Performance Indicator (KPI) is frequently used to refine and choose the best models. If alternative circumstances arise at test time, such as a shift in class priors (prior shift), the accuracy may significantly decline. A probabilistic predictor can be readily adapted to the new priors when the test time priors are known, but if the predicted probabilities are off, this approach may fail (or incur a performance cost). The probabilities of all classes count in this situation, as opposed to the scenarios

discussed before, when only the confidence of the anticipated (most probable) class mattered. Particularly, their estimated probability must be in the proper odds for correct performance as unusual classes grow more prevalent.

- A decision system based on probabilistic predictors is used to suggest a course of therapy or a different diagnosis once the test data, such as clinical data, is provided. In the future, the whole cost of the therapy will be disclosed (after more information about each case is gradually collected). The ability to predict the activities' anticipated costs in advance is crucial. It is extremely undesirable when the actual costs consistently end up being substantially greater than those predicted. Correctly estimating the likelihood of making mistakes, with a focus on those that result in high costs, is necessary to accurately predict projected treatment costs.

- In a situation requiring cost-sensitive decision making, the trained predictor is employed. Adopting a trained probabilistic predictor for a given cost (loss) matrix is simple when utilizing the Bayesian optimal strategy. When opposed to identifying the most likely state and deciding in light of it, the costs may underline the significance of certain sorts of errors and significantly influence the decisions. However, because this technique uses all predicted probabilities, it could not work well if the probabilities are off.

1.2.2 When Probabilistic Predictions Can Become Misleading

- As a result of overfitting the models during training.

- The predictor performance may significantly deteriorate if the appearance of the classes changes (prior shift) or even if a greater qualitative change in the input occurs (out of domain samples). Methods for tackling classifier adaptation exist, however, it is not guaranteed that they will ensure reliable probabilistic predictions.

- As a result of some perturbations, e.g. slight adversarial perturbations can turn over the predictions with a high confidence.

1.2.3 How We Can Improve Calibration

- Trained predictors that generalize well with negative log-likelihood are often better calibrated. Large-scale supervised training or large-scale self-supervised training, as well as fine-tuning for the target dataset, are common training methods that are costly. The accuracy is the KPI in supervised training, and the training processes are optimized for greater generalization. It is preferable to avoid complete retraining since in this situation, since the reliability of predicted probabilities is a secondary aim.

- One can learn a parametric or non-parametric correction of the probabilistic predictor using a trained model and an independent validation (calibration) labeled dataset. We will refer to this as post-hoc calibration (or post-training re-calibration). It is unclear what calibration criteria are pertinent and how to assess and compare calibration procedures if one does not know why calibration is required. Because of this, we must make some extra assumptions about the downstream task, at least for the purposes of assessing and contrasting calibrating approaches.

1.2.4 Scenarios we Focus on

Based on the aforementioned study, we find two crucial downstream tasks that are impacted by calibration: prior shift adaptation and Bayesian decision making in a cost-sensitive environment, with the first issue being more general. We note the following situations as being of practical concern:

- At the time of calibration, the cost matrix or a prior shift is already known. The objective is to change the predictor's predictive probabilities so that the risk (expected cost) of the predictions is reduced.
- At calibration time, the loss matrix is known for one scalar parameter. For instance, we may be aware of a state's hazard and high cost of error, but the precise amount of this cost won't be known until a test. Another illustration is confidence calibration, where it is known that at test time a recognition with a reject option will be employed, but the rejection cut-off cost is unknown in advance.
- At calibration time, prior shift is known for one scalar parameter. Possible examples include the linear interpolation of the priors between two known distributions in one dimension (e.g. between two different seasons, or two different geographical locations). Another frequent scenario is when someone has a long-tailed training and calibration data but wants to get a high performance (accuracy) for both the original data and data with uniformly distributed classes (emphasizing the performance on rare classes). In this

instance, we may think of the long-tailed distribution and the uniform distribution of priors as a convex 1-parameter family.

- It is assumed that the loss matrix or a prior shift belongs to a more general set that is defined at calibration time. The use cases stated above are obviously expanded by this.

1.3 Objectives

In this thesis, we try to calibrate our predictions for a specific downstream decision-maker, taking inspiration from (Zhao et al., 2021). We study different notions of calibration, measures of miscalibration, including hypothesis testing. Further, we note that most neural networks are optimized using the cross-entropy loss, which is used because of it's nice properties as a surrogate to the 0/1 loss (Bartlett et al., 2006). However, many tasks do not have a 0/1 loss. Because of that, many new approaches for such tasks are being developed. One of which would be the calibration for decision-making, where one would try to calibrate so that the new predictions take into account the non–0/1 loss. Unlike (Zhao et al., 2021), where the calibration is done with respect to a set of losses, we try to calibrate with respect to a specific loss matrix. To achieve that, we took inspiration from (Song et al., 2016), where a form of direct empirical risk minimization was formed. From that, we designed a loss criterion referred to as *Direct Loss*. Further, we also propose another loss criterion referred to as *Integral Loss*, which allows to optimize for a convex combination of two loss matrices. We explain our thought process of the derivation of the criterions, derive some of the theoretical properties, and experimentally show the performance.

1.4 Thesis structure

In chapter 2 we provide an overview of the technical background knowledge concerning the notation, the different calibration strategies and methods. Chapter 3 includes our new proposed loss criterions and explains them and the motivation behind them. Chapter 4 then describes and discusses the experiments performed using our new loss criterions and the current existing loss criterions (e.g. Negative Log-likelihood, followed up by a discussion of the results. Finally, we end with the concluding chapter 5, where we summarize the work we have done.

1.5 Author Contribution Statement

Methods and theoretical claims were proposed by the supervisor and proven by the author (with varied degrees of help), implementations and experiments were performed by the author. Some preliminary results were obtained during the semestral project initially supervised by prof. Jiří Matas and some results were presented in the paper submission for ICLR 2023 [https://openreview.net/forum?id=ih3mo7J-vb].

Chapter 2

Background

In this chapter, we summarize the basic notations, definitions and concepts relevant for our topic. For the sake of completeness, we also include topics that we do not directly use, however are a good introduction into the topic of calibration for the reader.

2.1 Probabilistic settings

Let $\{(X_i, Y_i)\}_{i=1}^{n}$ be independent and identically distributed random variable pairs from $\mathcal{X} \times \mathcal{Y}$, where $\mathcal{X}$ is the input space and $\mathcal{Y}$ is the finite set of m class labels.

Consider $\Delta^m = \{x | x \in \mathbb{R}^m_{\geq 0} \wedge ||x||_1 = 1\}$ – in other words, the $(m-1)$-dimensional probability simplex. We then define our probabilistic model as $g : \mathcal{X} \to \Delta^m$. In other words, for every input x, g outputs a probability vector that would reflect the following meaning if it was perfectly calibrated:

$$(\mathbb{P}[Y = 1 \mid X = x], \ldots, \mathbb{P}[Y = m \mid X = x])$$

2.2 Classification

Consider having finite dataset τ coming from the distribution $P(X, Y)$. Classification consists of learning a mapping $f : \mathcal{X} \to \mathcal{Y}$, that is an assignment of a class label to each input. This is often done in the way of predicting the probabilities of each class label assignment, similarly to the definition of a probabilistic model g defined above. From the predicted probabilities, one can have strategies to choose the final predicted class label. The most widely used strategy is to take the class with the highest probability. Thus, the learned mappings f make their prediction by predicting the most confident class using the probabilistic model g:

$$f(x) = \arg\max_y g_y(x).$$

From there, under the Empirical Risk Minimization (ERM) framework, one would like to minimize the expected loss of the model, that is:

$$R(f) = \mathbb{E}_{(x,y)\sim P(X,Y)}[l(f(x), y)], \tag{2.1}$$

where $l : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ is a chosen loss function. Not having $P(X, Y)$, the standard practice is to estimate the risk empirically, using an average over the data.

$$\hat{R}(f) = \frac{1}{|\tau|} \sum_{(x,y)\in\tau} l(f(x), y). \tag{2.2}$$

From there, if l is the 0/1 loss (matrix), i.e:

$$l(\hat{y}, y) = \begin{cases} 0 & y = \hat{y} \text{ (correct prediction)} \\ 1 & y \neq \hat{y} \text{ (wrong prediction)} \end{cases}$$

we arrive at the widely used accuracy metric, i.e. the fraction of predictions the model classified correctly.

In practice, the model g is parametrized with θ. These parameters θ are usually estimated using the maximum likelihood estimate. Further, the 0/1 loss is not tractable as it is not differentiable, thus surrogate losses are being used. Given the surrogate loss, we expect the parameters θ to be statistically consistent, i.e. as the number of the samples increases, θ will converge in probability to the true value. Sadly, these parametrized models miss-specified in the sense that they are often not in the hypothesis family of the task, as the dynamics of the real world tend to be more complex. To overcome that, we tend to overparametrize the neural networks and therefore cannot hope to rely on statistical consistency – as the number of parameters increase, the sample size has to be even greater.

2.3 Neural Networks as probabilistic classifiers

Unsurprisingly, the concept of neural networks first emerged as a model of how we thought brain neurons behave. This model, known as ”connectionism”, used connected circuits to imitate intelligent behavior. Nowadays, neural networks have become the model of choice for classification in many domains, especially in computer vision and natural language processing.

Neural networks as classifiers output a probability prediction vector $\hat{\mathbf{p}}$ for each input $\mathbf{x}$. The probability $\hat{\mathbf{p}}$ is computed as $\hat{\mathbf{p}} = \sigma(\mathbf{z})$, where the softmax function σ takes the form of:

$$\sigma(\mathbf{z})_k = \frac{\exp(\mathbf{z}_k)}{\sum_{j=1}^{m} \exp(\mathbf{z}_j)}, \tag{2.3}$$

and $\mathbf{z}$ are called the logits of the network, i.e. the nonprobabilistic outputs of the last layer before applying the softmax function. From there, the predicted class label for a given probability prediction vector is computed as:

$$\hat{y} = \arg\max_k \hat{\mathbf{p}}_k. \tag{2.4}$$

They are typically trained using gradient descent methods optimizing a certain loss criterion. As equation (2.2) for the 0/1 loss is neither differentiable nor provides informative gradients, we have to opt for different optimization criterions. For classification, the standard loss function is the NLL. NLL takes as it's input the logits and the one hot encoded true label $\mathbf{y}$ and calculates the loss in the following way:

$$\mathrm{NLL}(\mathbf{y}, \mathbf{z}) = -\log \hat{\mathbf{p}}_y = -\sum_{k=1}^{m} \mathbf{y}_k \log(\sigma(\mathbf{z})_k). \tag{2.5}$$

As $\mathbf{y}_k \in \{0, 1\}$, and $\mathbf{y}$ is a one hot encoded vector, the sum essentially finds the correct label and the value $-\log(\sigma(\mathbf{z})_k)$ is going to be the loss. We visualize the NLL function compared to the 0/1 loss in figure 2.a. As the probability for the correct class increases, the loss decreases and vice versa.

NLL comes from the maximum likelihood estimation principle, where one tends to minimize the negative logarithm of the conditional probabilities given parameters. Unlike the 0/1 loss, NLL is differentiable, convex and provides informative gradients. Having a trained neural network, the outputs are interpreted as probabilities, even though in many cases they are not, which causes problems in decision making in downstream tasks, where the interpretability is an important factor. To that end, the field of calibrating probabilistic outputs has been gaining popularity.

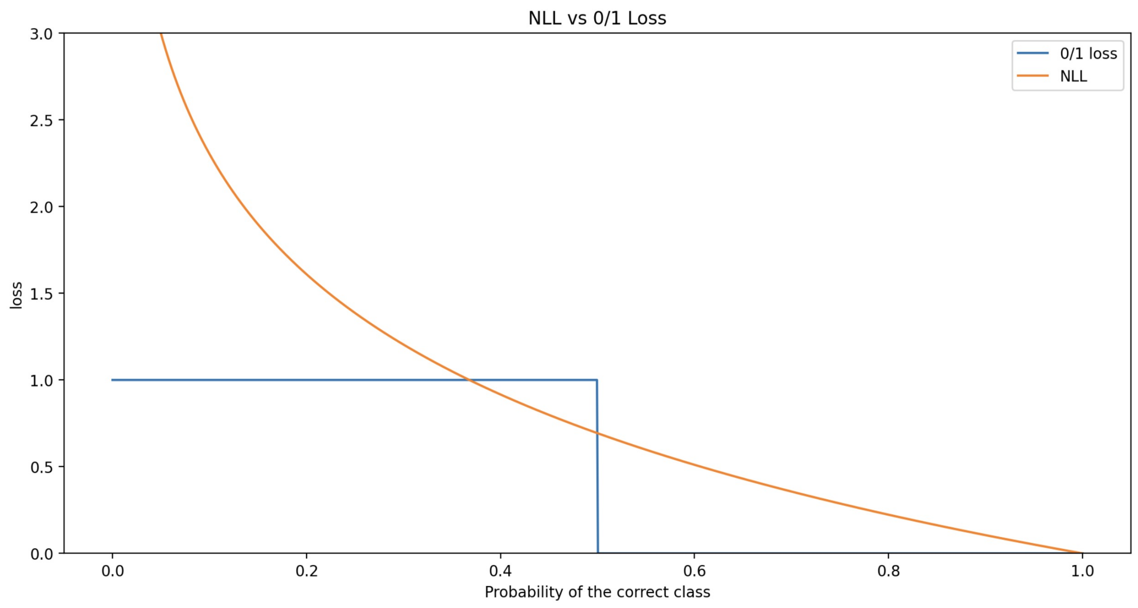

Figure 2.a: The NLL function for different input probabilities of the true class in a binary classification problem compared to the 0/1 loss. The x-axis shows the probability of the true class and the y-axis shows the loss given the probability.

2.4 Calibration definitions

We summarize three different definitions of calibration. The first calibration definition states that a model is calibrated if

$$\mathbb{P}[Y = \arg\max\ g(X) \mid \max\ g(X) = z] = z \tag{2.6}$$

for $z \in [0, 1]$, i.e. if the predicted probability most confident prediction is calibrated then we say that the model is calibrated. To avoid confusion, if a model meets this definition, we say that the model is *confidence calibrated*. Many works are concerned only with this definition and call models that fulfill this definition as fully calibrated, i.e. unaware of stronger definitions (Guo et al., 2017) as for 0/1 loss matrices, the class with the highest probability is the best prediction. Note that it is sufficient to supply predictions with a reliable confidence, which solves the issue of overconfident wrong predictions.

A stronger definition discussed by (Zadrozny & Elkan, 2002) says it is also desired that the predictions are calibrated in each of the classes, that is

$$\mathbb{P}[Y = y \mid g_y(X) = z] = z, \tag{2.7}$$

for all $y \in \{1, 2, \dots, m\}$ and $z \in [0, 1]$. We call a model calibrated according to (2.7) as *class-wise calibrated*.

Lastly, the strongest definition considered in (Vaicenavicius et al., 2019) requires that the distributions of target classes given an input is equal to the prediction. Note, this definition was known much earlier in the forecasting literature (Bröcker, 2009). Mathematically, it reads as

$$\mathbb{P}[Y = y \mid g(X) = \mathbf{p}] = \mathbf{p}_y, \tag{2.8}$$

for all $y \in \{1, 2, \ldots, m\}$ and $\mathbf{p} \in \Delta^m$. We reference models calibrated according to (2.8) as *strongly calibrated* or *distribution calibrated.* As one could guess, conditioning on the whole probability vector makes the calibration process very data demanding, since it is sensitive to hyperparameters such as the number of bins, which will be explained later.

2.4.1 Toy example

As a toy example to grasp the notion of different calibration definitions we present an example from (Vaicenavicius et al., 2019) in figure 2.b. Consider all the predictions that our model outputs to be the left column of the table and the true conditional distribution $\mathbb{P}[Y = y \mid g(X)]$ for $y \in \{1, 2, 3\}$ as the second column. Then the classifier g would be calibrated weakly, class-wise but not strongly.

$g(X)$	$\mathbb{P}[Y = y \mid g(X)]$
(0.1, 0.3, 0.6)	(0.2, 0.2, 0.6)
(0.1, 0.6, 0.3)	(0.0, 0.7, 0.3)
(0.3, 0.1, 0.6)	(0.2, 0.2, 0.6)
(0.3, 0.6, 0.1)	(0.4, 0.5, 0.1)
(0.6, 0.1, 0.3)	(0.7, 0.0, 0.3)
(0.6, 0.3, 0.1)	(0.5, 0.4, 0.1)

Figure 2.b: A toy example taken from (Vaicenavicius et al., 2019) for understanding the calibration definitions. The right column is the true conditional distribution, the left column is are the models predictions. Then g is calibrated by (2.6) and (2.7) but not calibrated by (2.8).

2.5 Calibration measures

To measure calibration, consider the calibration function $r : \Delta^m \to \Delta^m$ as

$$r(\boldsymbol{\mu}) = (\mathbb{P}[Y = 1 \mid g(X) = \boldsymbol{\mu}], \dots, \mathbb{P}[Y = m \mid g(X) = \boldsymbol{\mu}]) \tag{2.9}$$

for $\boldsymbol{\mu} \in \Delta^m$ – that is, how $\boldsymbol{\mu}$ would look if it were strongly calibrated. This implies that a model g is strongly calibrated if

$$r(g(X)) - g(X) = 0 \tag{2.10}$$

This also tells us how much $g(X)$ deviates from a perfectly calibrated model. A fair remark is that although usually omitted, equalities for equations including random variables have the meaning of being almost surely equal, i.e. equal with respect to the events that are not of measure zero. Given a distance function $d : \Delta^m \times \Delta^m \to [0, \infty)$ the Expected Calibration Error (CE) is defined as

$$\mathrm{CE}_d = \mathbb{E}[d(r(g(X)), g(X))]. \tag{2.11}$$

If we consider $A \subseteq \Delta^m$, a subset of the possible predictions of our model g, we could also use a different aggregating function, such as

$$\mathrm{CE}_d = \max_{\mu \in A}[d(r(\mu), \mu)], \tag{2.12}$$

depending on the use case.

2.5.1 Empirical Expected Calibration Error

To approximate CE a measure widely used for the weak calibration denoted as the ECE (Naeini et al., 2015) discretizes the probability interval into bins, assigns each prediction into a bin and then computes the average difference between the average confidence and the accuracy per bin weighted by the number of samples in the bin:

$$\mathrm{ECE} = \frac{1}{N} \sum_{b=1}^{B} n_b \, |\mathrm{acc}(b) - \mathrm{conf}(b)|, \tag{2.13}$$

where B is the number bins, N is the number of predictions, n_b the number of samples in bin b, $acc(b)$ the average accuracy of bin b and $conf(b)$ the average confidence in bin b. As discussed in (Vaicenavicius et al., 2019), (2.13) is the CE in an induced binary problem with the total variation distance(TV) $d(x, y) = \frac{1}{2}||x-y||_1$. This measure corresponds to the definition in (2.6). Note that having 0 ECE does not imply a good model, as even a coin flip model could have 0 ECE while having a

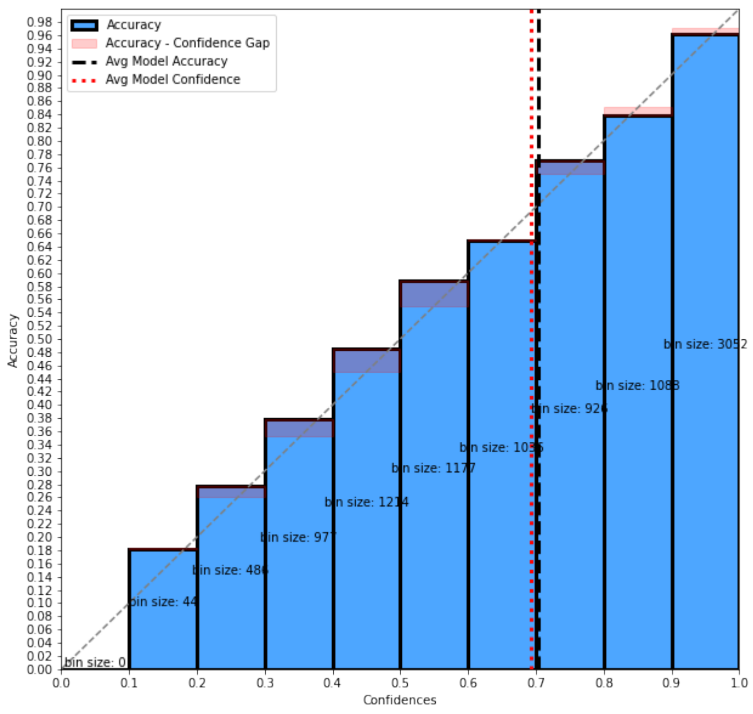

Figure 2.c: A reliability diagram of a classifier with 70% accuracy. The model is slightly overconfident in the high confidence bins, and slightly underconfident in the medium confidence bins. Image generated from the predictions of a LeNet architecture (LeCun et al., 1998) trained on a CIFAR-10 (Krizhevsky et al., 2009) dataset.

50% accuracy, i.e. calibration does not imply accuracy. The possible pitfall of this metric is the parametrization of the number of bins.

The ECE is nicely visualized using the so-called reliability diagrams. Since for every data sample we consider only the most confident prediction, we can visualize these in a histogram, binned by confidences. By definition (2.6), we expect the histogram to follow the line $y = x$. We show an example of a reliability diagram in figure 2.c.

2.5.2 Static Calibration Error

As ECE takes into account only the highest confidence per data point, a measure for the class-wise definition (2.7) was derived as a generalization of ECE. Static Calibration Error (SCE) (Nixon et al., 2019) was proposed as

$$\mathrm{SCE} = \frac{1}{K}\sum_{k=1}^{K}\frac{1}{N}\sum_{b=1}^{B} n_{bk}\,|\mathrm{acc}(b,k) - \mathrm{conf}(b,k)|, \tag{2.14}$$

where K is the number of classes. It is easy to notice that SCE is just an average ECE over classes.

2.5.3 Strongest ECE

To measure strong calibration according to definition (2.8), the derivation comes as an estimate of the basic form of the CE in (2.11). Proposed in (Vaicenavicius et al., 2019), assume a binning $\{\mathbf{b_i}\}_{i=1}^{B}$ of Δ^m and denote the bin containing a vector $w \in \Delta^m$ by $\mathbf{b}[w]$. Define the empirical estimate of the calibration function $\hat{r} : \Delta^m \to \Delta^m \cup \{0\}$ for the y-th class as

$$\hat{r}(w)(y) = \frac{|\{i : g(X_i) \in \mathbf{b}[w] \wedge Y_i = y\}|}{|\{i : g(X_i) \in \mathbf{b}[w]\}|} \tag{2.15}$$

for all $w \in \Delta^m$ and $\hat{r}(w) = 0$ if the denominator is zero – the bin $\mathbf{b}[w]$ is empty. With this definition, given an observed dataset, as $\hat{r}$ is a piecewise constant on every bin $\mathbf{b_i}$, we can define $\hat{r}_i$ as the strongly calibrated prediction for the bin $\mathbf{b_i}$.

Next, for all bins $\mathbf{b_i}$ we consider the average prediction $\hat{g}_i$ and the fraction of the bin $\hat{p}_i$ – the weight, calculated as

$$\hat{g}_i = \frac{\sum_{j:g(X_j)\in\mathbf{b_i}} g(X_j)}{|\{j : g(X_j) \in \mathbf{b_i}\}|}$$

and

$$\hat{p}_i = \frac{|\{j : g(X_j) \in \mathbf{b_i}\}|}{|\{j : g(X_j) \in \Delta^m\}|},$$

respectively. Define the estimates as zero if the denominators are zero.

From that, we can define the measure of miscalibration $\widehat{CE}_d$ in terms of expectancy given a distance function as

$$\widehat{\mathrm{CE}}_d = \sum_{i=1}^{B} \hat{p}_i d(\hat{r}_i, \hat{g}_i) \tag{2.16}$$

It was also shown in (Vaicenavicius et al., 2019) that this estimator converges to CE_d as the number of bins and the number of data samples grow to infinity (under some additional assumptions, like Lipschitz continuity).

2.5.4 Maximum Mean Calibration Error

We also only mention another measure of miscalibration referred to as the Maximum Mean Calibration Error (MMCE). It is proposed by (Kumar et al., 2018) to tackle the problem of ECE not being differentiable everywhere. The idea is to calculate the calibration error using kernels.

2.5.5 Binning options

In practice, one has to discretize the probability simplex into bins, therefore we show the main hyperparameters of binning options.

Number of bins

Intuitively, the higher the number of bins, the less data per bin and thus the higher variance of the estimates in each bin.

Data-independant binning

The simplest option of binning is to bin the probability simplex into equal sized bins, which is also often done in the machine learning literature for binary problems e.g. (Guo et al., 2017). For one dimension ($m = 2$), we split the probability interval. For multiple dimensions ($m > 2$) we split the probability interval along each dimension of the simplex, meaning we grow exponentially in the number of dimensions.

Data-dependant binning

Data-dependant binnings allow us to create binnings with desired features such as the minimum number of samples per bin by thresholding. One way to create a data-dependant binning is to iteratively choose a splitting dimension that has the maximal variance and then split this dimension based on a central value method such as the average or median. This iterates until there are no more dimensions to split due to the allowed minimal size of the bin. The advantage of this method is that certain estimators converge faster to the real value, e.g. the calibration function (2.15) (Nobel, 1996), as the estimators are dependant on the binnings. However, one is also prone to overfitting.

2.5.6 Comparing calibration error estimates

Taken as an example from (Vaicenavicius et al., 2019), consider two probabilistic models g and g' and let $\widehat{\mathrm{CE}}_d$ and $\widehat{\mathrm{CE}'}_d$ be their estimate of CE_d, respectively. The common practice would be to compare these two estimates and then based on the comparison make an inference about the calibration of the model. However, as the biases of the estimators may differ, this comparison may be unjustified.

Consider a binary classification problem that is clearly separable and has equal prior probabilities, that is $p(Y = 1) = p(Y = 2) = 1/2$. A constant model g^{const} and the optimal model g^{opt} are calibrated, therefore their expected calibration error is zero.

If we define the estimator $\widehat{\mathrm{CE}}_{TV}$ for only one bin and consider only one input, the bias of the perfect model will be $\mathbb{E}[\widehat{\mathrm{CE}}_{TV}] - \mathrm{CE}_{TV} = 0$. Yet, for the constant model, this bias will be $1/2$ as the expected calibration error is always $1/2$. Comparing these two errors leads us to a wrong conclusion that the constant model is less calibrated than the perfect model, even though that is not the case here. Hence, one would have to use a hypothesis testing approach, which accounts for the different biases and the fact that $\widehat{\mathrm{CE}}_d$ and $\widehat{\mathrm{CE}'}_d$ are random variables. Consider now the case where we would have unbiased estimators. Still, the comparison may not be justified if the estimators for each model have different distributions as they are random variables.

2.5.7 Consistency resampling

Firstly, a resampling technique referred to as *consistency resampling* (Bröcker & Smith, 2007) is employed. The idea behind it is to compare a bootstrapped distribution of $\widehat{\mathrm{CE}}_d$ with a bootstrapped distribution of a perfectly calibrated predictor $\widehat{\mathrm{CE}}_d^{id}$, which we create by using artificial labels. Let us now illustrate the process for a binary classification problem.

A single resampling cycle consists of the following steps. Consider our set of predictions $\{g(x_i)\}_{i=1}^N$ and the set $\{z_i\}_{i=1}^N$ of independent uniformly distributed random variables.

1. Sample with replacement N times from the set $\{g(x_i)\}_{i=1}^N$, obtaining the proxy set $\{\widehat{g(x_i)}\}_{i=1}^N$.
2. Create proxy labels $\{\hat{Y}_i\}_{i=1}^N$ according to

$$\hat{y}_i = \begin{cases} 1 & z_i < \widehat{g(x_i)} \\ 0 & \text{otherwise} \end{cases}$$

By definition, the proxy dataset $\{\widehat{g(x_i)}, \hat{y}_i)\}_{i=1}^N$ will be calibrated: $\mathbb{P}(Z_i = 1|\widehat{g(x_i)}) = \widehat{g(x_i)}$. This resampling step will be computed N_{boot} times, creating datasets from which we can calculate the estimators $\widehat{\mathrm{CE}}_d$ and approximate $\widehat{\mathrm{CE}}_d^{id}$ – the distribution of the expected calibration error under the assumption that our model is calibrated. The resampling technique can be generalized to multiclass problems by comparing z_i with the cumulative sum over $\widehat{g(x_i)}$.

2.5.8 Hypothesis Testing

As mentioned earlier, one can use the distribution $\widehat{\mathrm{CE}}_d^{id}$ obtained via consistency resampling to perform statistical hypothesis tests. The p-value would be in the form of $\mathbb{P}[\widehat{\mathrm{CE}}_d^{id} \geq \widehat{\mathrm{CE}}_d]$, that is the probability of our estimate of the expected miscalibration being smaller than the miscalibration error of a calibrated model. Comparing these probabilities for two different models and the chosen significance level would allow us to choose a more calibrated model without having to worry about the different biases of the estimates.

Unfortunately, in practice, we are limited by the power of these statistical test with the amount of data we can gather. Having error approximations from binning the probabilities and approximation errors from bootstrapping makes the tests reject the null hypothesis that the model is calibrated too often (Widmann et al., 2019).

2.6 Methods for calibration

In the literature we found that multiple approaches for calibration are currently being researched. Approaches differ based on the time of calibration and loss awareness. Also, one would want the calibration to preserve the accuracy (therefore only changing the predicted probability so that the ranking stays the same) and to be data-efficient – little data is available for the calibration and evaluation. Some approaches try to directly train a calibrated neural network (Kumar et al., 2018), while others look for an algorithm to calibrate the outputs of the neural network (Guo et al. 2017; Alexandari et al. 2020; Sahoo et al. 2021; Kull et al. 2019). The loss awareness is mostly differentiated between tasks, where the 0/1 loss is assumed and tasks where one tries to calibrate for a set of loss matrices (Zhao et al., 2021). Here, we provide a short summary of some of the methods.

2.6.1 Recalibrating an uncalibrated Neural Network

Also referred to as post-hoc calibration, takes as it's input either the predicted logits or probabilities from a trained model in a semi–supervised or supervised way and finetunes/transforms them, so that the resulting probabilities will be calibrated.

One approach is to parametrize the logits, so that the resulting probabilities after the softmax function become more reliable. Another one is to correct the resulting probabilities into more reliable probabilities by creating a mapping. These methods also differ by the optimization criterion, which tends to be NLL and are trained on a held-out validation/calibration dataset. Figure 2.d visualizes these steps in a diagram. As mentioned earlier, using NLL is not accounting for a different loss than 0/1. We now review the basic parametrization methods for calibration.

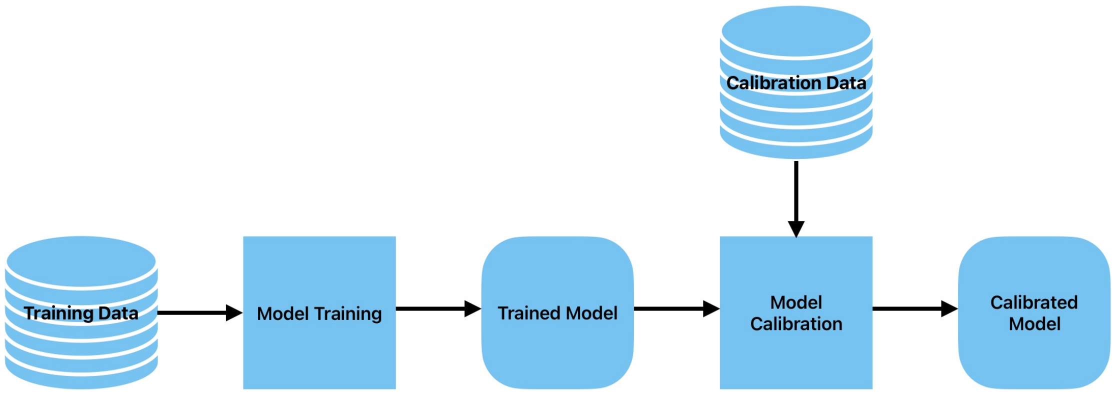

Figure 2.d: The general process of post-hoc calibration. The model is calibrated after the actual training.

Platt Scaling

Introduced in (Platt et al., 1999) as an improvement for Support Vector Machines, the Platt Scaling method for a binary probabilistic classifier learns scalar parameters $a, b \in \mathbb{R}$ – the calibrated probabilities will take the form of $\sigma(a\mathbf{z}_1 + b)$.

Temperature Scaling (TS)

Proposed in (Guo et al., 2017), TS was a novel way of calibrating the most confident predictions. The idea is quite learning a single parameter $T \in \mathbb{R}_{>0}$ that we use as $\sigma(\mathbf{z}_i/T)$. As $T \to \infty$ the distribution transforms into a uniform distribution and as $T \to 0$ the probability of the most confident class will be equal to 1. Note that, as it is a monotonic operation, the accuracy of the model does not change, while the confidence scores do.

Bias-Corrected Temperature Scaling (BCTS)

In (Alexandari et al., 2020) it was experimentally found that TS does not correct the systematic bias in the estimates of the priors under the model and therefore proposed to not only learn the parameter $T \in \mathbb{R}$, but also a vector $\mathbf{b} \in \mathbb{R}^m$ so that every class has its own bias term, i.e. $\sigma(\mathbf{z}_i/T + \mathbf{b})$. This is referred to as BCTS. It is a simple mutliclass extension of Platt Scaling.

Vector Scaling (VS) / Matrix Scaling (MS)

VS and Matrix Scaling (MS) were introduced as a further extensions of Platt Scaling. MS takes the form of $\sigma(\mathbf{W}^T\mathbf{z}_i + \mathbf{b})$, where $\mathbf{b} \in \mathbb{R}^m, \mathbf{W} \in \mathbb{R}^{m,m}$. VS is a special case of MS for which the matrix $\mathbf{W}$ is a diagonal matrix. Note that for matrix scaling the number of parameters grows quadratically with the number of classes.

Optimizing NLL instead of ECE?

The standard practice to train these parametrizations is to optimize them with respect to NLL (Guo et al. 2017; Alexandari et al. 2020). The curious reader might have asked themselves why should the calibration methods minimizing NLL improve measures of miscalibration such as ECE? Why not directly optimize ECE, as though it is not differentiable everywhere, the set of non-differentiable points is of measure zero. For the full proof, we refer the reader to (DeGroot & Fienberg 1981; Bröcker 2009; Murphy & Winkler 1987; Gneiting & Raftery 2007, where it is formally proven why minimizing NLL instead of ECE is perfectly justifiable. In short, NLL corresponds to a proper scoring rule and therefore is decomposable into parts that include both uncertainty and reliability. A calibrated predictor's reliability will be 0. Minimizing NLL with respect to the calibration will decrease the reliability term therefore minimizing the miscalibration. As a result, we designed a loss function that corresponds to a proper scoring rule and is based on ERM, allowing us to minimize miscalibration using the ERM framework.

2.6.2 Training a calibrated Neural Network

In (Kumar et al., 2018) the following is suggested. Given a dataset D, the idea to train a neural network that is better calibrated is to solve the following:

$$\min_{\theta} \text{NLL}(D, \theta) + \lambda \text{CE}(D, \theta), \tag{2.17}$$

where θ are the neural networks parameters, NLL is the negative log-likehood, CE is a calibration error measure and λ is the importance weight of the measure. To calibrate the strongest prediction score, one could think about using the ECE as the CE, but with the hopes of not getting numerical instabilities of the non-differentiable point. In the paper, the metric MMCE is used. One can later combine this approach with the recalibration after training approach as well.

2.6.3 Calibrating for decision-makers

Downstream decision-makers use the confidence of the predictions. For that, one would want distribution calibrated outputs, however as mentioned earlier, these are hard to achieve, since it requires exponential sample complexity in the number of classes. This problem lead to a relaxed calibration definition referred to as *Decision Calibration*. In simple terms, decision calibrating means to calibrate so that the risks under the predicted distribution does not differ from the risks under the true distribution. Note that as decision-makers one tries to incorporate the loss function into the process, therefore this is a loss aware task. We now provide a formal definition of this concept.

Bayes Optimal Decision

Consider a finite action space $\mathcal{A}$. Let us now redefine the loss matrix to be $l : \mathcal{Y} \times \mathcal{A} \to \mathbb{R}$. This terminology is only defined to fit the decision-making framework, however could be easily mapped to classification tasks by considering $\mathcal{A} = \mathcal{Y}$. Let $p^*(k|x)$ be the true conditional probability of a sample $x \in \mathcal{X}$ to be of class k and $\hat{p}(k|x)$ the predicted ones. We define the *Bayes Optimal Decision* strategy $q : \Delta \to \mathcal{A}$ given the predictive conditional probability to be:

$$q_l(p(\cdot|x)) = q(p(\cdot|x)) = \underset{d\in\mathcal{A}}{\arg\min} \sum_{k\in\mathcal{Y}} p(k|x)l(k,d). \tag{2.18}$$

Strategies that are Bayes optimal decision strategy with respect to the loss matrix l will be referred to as q_l. Naturally, the definition of the risk of any strategy q comes up as:

$$R(q) = \mathbb{E}[l(Y, q(p(\cdot|X))]. \tag{2.19}$$

Trust would not be an issue if the forecaster were able to predict optimal forecasts (i.e., $\hat{p}(X) = p^*(X)$), but since this is rarely the case, the forecaster must find workable strategies to inspire confidence in the decision-makers, which leads to the following definition.

Decision Calibration (Zhao et al., 2021)

For any set of loss functions $\mathcal{L} \subset \mathcal{L}_{all}$, where $\mathcal{L}_{all} = \{l : \mathcal{Y} \times \mathcal{A} \to \mathbb{R}\}$ (that is all loss functions) and a set of strategies $\mathcal{Q} \subset \mathcal{Q}_{all} := \{q : \Delta \to \mathcal{A}\}$, we say that a prediction $\hat{p}$ is $(\mathcal{L}; \mathcal{Q})$-decision calibrated (with respect to p^*) if $\forall l \in \mathcal{L}$ and $\forall q \in \mathcal{Q}$ it holds that:

$$\mathbb{E}_X \mathbb{E}_{\hat{Y} \sim \hat{p}(\cdot|X)}[l(\hat{Y}, q(\hat{p}(\cdot|X)))] = \mathbb{E}_X \mathbb{E}_{Y \sim p^*(\cdot|X)}[l(Y, q(\hat{p}(\cdot|X)))]. \tag{2.20}$$

We refer to the left-hand side part of the equation as the **model risk** and the right-hand side as the **true risk**, reflecting the distributions used by the expectations. The model risk is the loss that the decision maker computes knowing only the input features and mimics the loss where the labels are drawn from the predicted distribution. The true risk is the true loss, without any mimicking. This captures the idea that the losses and the decision rules are not different between the predicted and the true distribution.

It is also shown that there exist $\mathcal{L} \in \mathcal{L}_{all}$, such that the definition 2.6.3 will coincide with any of the definitions mentioned in the section 2.4. For more details regarding the definitions, we encourage the reader to the original paper (Zhao et al., 2021). Furthermore, $\mathcal{L}$-decision calibration captures a desiderata, one would naturally expect in valid decision making.

Proposition 1. *(No regret, (Zhao et al., 2021)) Let the prediction function $\hat{p}$ be $(\mathcal{L}; \mathcal{Q})$-decision calibrated. Let $\mathcal{Q}_{\mathcal{L}}$ be the set of all Bayes Optimal Decision strategies with respect to loss functions in $\mathcal{L}$, i.e. $\mathcal{Q}_{\mathcal{L}} = \{q_l | l \in \mathcal{L}\}$. Then $\forall q' \in \mathcal{Q}_{\mathcal{L}}$ it holds that:*

$$\mathbb{E}_X \mathbb{E}_{Y \sim p^*(\cdot|X)}[l(Y, q_l(\hat{p}(\cdot|X)))] \leq \mathbb{E}_X \mathbb{E}_{Y \sim p^*(\cdot|X)}[l(Y, q'(\hat{p}(\cdot|X)))].$$

It states that the Bayes optimal decision for the given loss l is not worse than any other Bayes optimal decision strategy in q_l.

However, as the $(\mathcal{L}; \mathcal{Q})$-decision calibration definition was too general – containing infinite action spaces – a very similar but stricter definition that assumes a finite action space was provided.

2.7 Dataset shift

The joint distribution of inputs $P_{train}(x, y)$ and outputs $P_{test}(x, y)$ sometimes varies between the training and test stages, creating the problematic scenario known as the dataset shift, i.e: $P_{train}(x, y) \neq P_{test}(x, y)$. Dataset shift occurs in most practical applications for a variety of reasons, including experimental design bias and the simple unreplicability of the testing circumstances during training. For instance, in a language classification assignment, the training data may have been taken in a lab setting with strict controls, but the test data may have been taken in a setting with more or less noise, e.g. bad microphone. Further for our purposes, it was shown that certain dataset shifts can be well estimated if we have calibrated probabilities (Alexandari et al., 2020).

As a result, this topic gained much attention recently in the research community. We briefly introduce some of the possible shifts that can occur.

2.7.1 Covariate shift

Covariate shift is defined as a problem for which the following two equations hold:

$$\begin{aligned} P_{train}(y|x) &= P_{test}(y|x) \\ P_{train}(x) &\neq P_{test}(x). \end{aligned} \tag{2.21}$$

This means that the relationship between the dependant and independent variables does not change, however the distributions of the independent variable changes. As an example, consider an image classification task where we classify animal types e.g. whether the image contains a dog or a bear or a cat etc. but the training set will omit some of the dog breeds in the testing set.

Under covariate shift, model selection techniques like Cross Validation (CV) do not work optimally, as they require the unbiasedness for the estimation of the score of CV (Sugiyama et al., 2007).

2.7.2 Prior shift

Prior shift, as the name suggests, is concerned with the change of the prior probability of the dependant variable, i.e.:

$$\begin{aligned} P_{train}(x|y) &= P_{test}(x|y) \\ P_{train}(y) &\neq P_{test}(y). \end{aligned} \tag{2.22}$$

The interpretation here is much more obvious. Consider any classification task, but change the class wise distribution during the training and testing, e.g. dog vs. cat classification, but while the training dataset is going to have 50% dogs and 50% cats, the testing dataset is going to have 80% dogs and 20% cats. Further, we later show calibration can be sustained under the prior shift using importance reweighting.

2.7.3 Concept shift

Concept drift is the last missing piece of the shifts we summarized. Unlike prior shift or covariate shift, concept drift takes into account the conditional probability. It includes both

$$\begin{aligned} P_{train}(x|y) &\neq P_{test}(x|y) \\ P_{train}(y) &= P_{test}(y) \end{aligned} \tag{2.23}$$

and

$$\begin{aligned} P_{train}(y|x) &\neq P_{test}(y|x) \\ P_{train}(x) &= P_{test}(x). \end{aligned} \tag{2.24}$$

Easy examples of data that could have concept drifts are time series that are non-stationary. The usual specific example here is to use profit prediction trained on the data before the 2008 financial crisis and testing on data after the said crisis, e.g. 2011. Since the whole socio-economic changed, the relationship between the independant variable and the dependant variable changed for the testing set. Calibration does not work here, as the conditional probabilities are not assumes to be equal. So far, we do not know any means of recovering the predictive probabilities under the concept shift.

Chapter 3

Method

In this chapter we introduce our approach to calibrating with respect to a given loss matrix. We propose a loss criterion that optimizes for a convex combination of two different loss matrices. We introduce the loss criterions, derive certain aspects and further describe their use for prior shift.

3.1 Motivation

Consider a setup, where a decision-maker is given a neural network that has a high accuracy, however trained using NLL. At test time, the decision-maker is given an arbitrary loss matrix, based on which the decision-maker will be judged. They will therefore want to use the Bayes optimal decision rule based on the predictor $\hat{p}$:

$$\hat{q}(x) = q(\hat{p}(\cdot|x)) = \arg\min_{d \in \mathcal{A}} \sum_{k \in \mathcal{Y}} \hat{p}(k|x) l(k, d), \tag{3.1}$$

where $x \in \mathcal{X}$, $\hat{p}(k|x)$ is the conditional probability for class k as an output from the neural network and $l(k, d)$ is the new loss matrix assigning a cost for choosing decision d while x being of class k. As the performance criterion is to minimize the empirical risk, i.e.

$$\hat{R}(\hat{q}) = \frac{1}{n} \sum_{i=1}^{n} l(y_i, \hat{q}(x_i)), \tag{3.2}$$

we would like to make the model and the calibration procedure to take into account the new given loss matrix. Consider $\hat{p}(k|x; \theta)$ to be the output of the neural network which was additionally parameterized for the purpose of calibration by θ in a similar fashion to either TS, BCTS or VS. We propose the following approach to calibration – Find the calibration giving the best performance in terms of the risks, which reads as:

$$\begin{aligned} \min_{\theta} \quad & \frac{1}{n}\sum_{i=1}^{n} l(y_i, q(\hat{p}(\cdot|x_i;\theta))) \\ \text{s.t.} \quad & q(\hat{p}(\cdot|x;\theta)) = \operatorname*{arg\,min}_{d\in A} \sum_{k\in\mathcal{Y}}^{m} \hat{p}(k|x;\theta) l(k,d). \end{aligned} \tag{3.3}$$

Unfortunately, this task is a bilevel optimization problem and therefore is strongly NP-hard (Hansen et al., 1992). As such, we will look for other methods to feasibly solve this task.

3.2 Analysis of Decision Calibration by Zhao et al., (2021)

First, we notice that the Decision Calibration of Zhao et al. (2021) is not sufficient for decision-making. Let us show an example of $\hat{p}$ that is decision calibrated, but a distribution calibration achivies a better risk.

Proof. Let the input space consist of only two elements $\mathcal{X} = \{x_1, x_2\}$, the action space be equal to the output space, which has also two classes, i.e. $\mathcal{A} = \mathcal{Y} = \{0, 1\}$. Now consider the predicted and true conditional probabilities to have these values:

$$\begin{aligned} \hat{p}(y=0|x_1) = 0.4,\ \hat{p}(y=1|x_1) = 0.6,\ \hat{p}(y=0|x_2) = 0.6,\ \hat{p}(y=1|x_2) = 0.4, \\ p^*(y=0|x_1) = 0.6,\ p^*(y=1|x_1) = 0.4,\ p^*(y=0|x_2) = 0.8,\ p^*(y=1|x_2) = 0.2. \end{aligned} \tag{3.4}$$

Further, let our $\mathcal{L} = \{f\}$, where f is the 0/1 loss matrix and $\mathcal{Q} = \{q_f\}$, that is the Bayes Optimal Decision function with respect to the 0/1 loss matrix f. Based on the 0/1 loss, the Bayesian Optimal Decision will be to predict the class with the higher probability, based on the predictions, meaning $q(\hat{p}(\cdot|x_1)) = 1$ and $q(\hat{p}(\cdot|x_2)) = 0$. Under these circumstances, the predictor $\hat{p}$ is $(\mathcal{L}; \mathcal{Q})$-decision calibrated, verified as follows.

The risk in the decision calibration definition given p and p' is calculated for our example as

$$\begin{aligned}\mathbb{E}_X \mathbb{E}_{Y\sim p(\cdot|X)}[l(Y, q(p'(\cdot|X)))] = \frac{1}{2}\big(&p(y=0|x_1)(f(0, q(p'(\cdot|x_1))))+\\ &p(y=1|x_1)(f(1, q(p'(\cdot|x_1))))+\\ &p(y=0|x_2)(f(0, q(p'(\cdot|x_2))))+\\ &p(y=1|x_2)(f(1, q(p'(\cdot|x_2))))\big).\end{aligned}$$

Since we have a 0/1 loss matrix, the model risk is $\frac{1}{2}(\hat{p}(y=0|x_1)+\hat{p}(y=1|x_2)) = 0.4$ and the true risk is $\frac{1}{2}(p^*(y=0|x_1)+p^*(y=1|x_2)) = 0.4$, proving the predictor is decision calibrated.

That is because decision calibration uses the predicted probability as the input to the Bayesian Optimal Decision strategy. If we were to use the true probability in the decision making, we achieve a risk of $\frac{1}{2}(p^*(y=1|x_1)+p^*(y=1|x_2)) = 0.3$. Not only do we get a smaller risk, but the strategy also changed under the 0/1 loss. $\square$

Having a calibration that would map accordingly, we could achieve that, however decision calibration does not allow that. In this sense, we would like to redefine the goal of calibrating for decision making.

Definition 1. *Let $r:\Delta\to\Delta$ be the distribution calibration mapping and $\hat{p}$ the probability predictor. The predictor $\hat{p}$ is considered to be calibrated if the true risk of the predictor $\hat{p}$ is equal to the true risk of the recalibrated predictor $r(\hat{p}(\cdot|x))$, which reads as:*

$$\mathbb{E}_X \mathbb{E}_{Y\sim p^*(\cdot|X)}[l(Y, q(\hat{p}(\cdot|X)))] = \mathbb{E}_X \mathbb{E}_{Y\sim p^*(\cdot|X)}[l(Y, q(r(\hat{p}(\cdot|X))))]. \tag{3.5}$$

Such predictors will be called truly decision calibrated (true coming from the fact that this time both the expectations consider the true risk).

Note that the left-hand side will always be greater or equal to the right-hand side, as the right-hand side uses perfectly calibrated probabilities. Therefore, the difference between the left-hand side and the right-hand side is a measure of miscalibration. We further know that minimizing the Empirical Risk using a post-hoc invertible calibration mapping is equivalent to minimizing this miscalibration (Ho et al., 2022), see (Gruber & Buettner, 2022) for general equivalence for proper scoring rules. We will thus look for ways to minimize the Empirical Risk using parametrizations in order to minizime this miscalibration.

We can see that a predictor that is decision calibrated by Zhao et al. (2021) is not necessarily truly decision calibrated, from the previous example. However, neither does a truly decision calibrated predictor need to be decision calibrated by Zhao et al. (2021).

Proposition 2. *Decision calibration by Zhao et al. (2021) is not necessary for decision-making.*

Proof. Again, consider the 0/1 loss matrix and some distribution calibrated predictor p^*, which is clearly truly decision calibrated. If we miscalibrate p^* using temperature scaling with a randomly chosen $T > 0$, the predictions do not reorder (the argmax remains equal), and thus under the 0/1 loss the miscalibrated predictor stays truly decision calibrated. However, for decision calibrated by Zhao et al. (2021), the model loss consists of an expectation over the predictions, therefore it may occur that the predictor is not going to be decision calibrated. $\square$

3.3 Direct Loss

For finite action spaces, one can see that the optimization criterion from the bilevel problem (3.3) is a piecewise constant function, consequently giving us gradients that are either zero or non-existent. The problem of optimizing piecewise constant functions was tackled in (Vlastelica et al., 2019). The method proposed is given a solver, using which one can minimize a piecewise constant function. The solver is formalized as a mapping that takes an input w and outputs a decision d based on the criteria:

$$Solver(w) = \arg\min_{d\in\mathcal{A}} w^T\phi(d) = \arg\min_{d\in\mathcal{A}} \sum_k w_k\phi_k(d), \tag{3.6}$$

where $\phi : A \to \mathbb{R}^n$ is an injective mapping of our decisions, e.g. one hot encoding. We can apply the following mappings to model Bayesian decision strategy:

1. Fix w_k as $\hat{p}(k|x)$, then $\phi_k(d) = l(k,d)$
2. Fix $\phi(d)$ as a one hot encoded vector of decision d, i.e. $\phi_k(d)$ equals one if d equals k and zero otherwise. Then $w_{d,k} = \hat{p}(k|x)l(k,d)$, in other words a matrix, where in each column indexed by d we have elements $\hat{p}(k|x)l(k,d)$ for $k = 1, 2, \ldots, m$.

Given the solver function and a piecewise constant loss L (in our case equation the Empirical Risk), in order to optimize our parameters, (Vlastelica et al., 2019) proposed to use algorithm 1 for calculating the gradient.

Algorithm 1: The general framework for calculating the gradient. (Vlastelica et al., 2019)

```
Function Forward($\hat{w}$):
    $\hat{y} := \mathrm{Solver}(\hat{w})$
    save $\hat{w}$ and $\hat{y}$ for backward pass
    return $\hat{y}$
Function Backward($\frac{dL}{d\phi(y)}(\phi(\hat{y}))$, $\lambda$):
    load $\hat{w}$ and $\hat{y}$ from forward pass
    $w' := \hat{w} + \lambda \cdot \frac{dL}{d\phi(y)}(\phi(\hat{y}))$ // nudge the parameters
    $y_\lambda := \mathrm{Solver}(w')$ // solve for the nudged parameters
    return $\nabla_w f_\lambda(\hat{w}) := -\frac{1}{\lambda}[\phi(\hat{y}) - \phi(y_\lambda)]$ // gradient of the continuous
     relaxation
```

Algorithm 2: The algorithm applied to case 1.

```
Function Forward($\hat{p}(\cdot|x), l$):
    $\hat{y} := \arg\min_d \sum_k \hat{p}(k|x) l(k,d)$
    save $\hat{p}(\cdot|x)$ and $\hat{y}$ for backward pass
    return $\hat{y}$
Function Backward($k^*$(the true label), $l$, $\lambda$):
    load $\hat{p}(\cdot|x)$ and $\hat{y}$ from forward pass
    $\hat{p}(\cdot|x)' := \hat{p}(\cdot|x) + \lambda\,\mathrm{one\_hot}(k^*)$
    $y_\lambda := \arg\min_d \sum_k \hat{p}(k|x)' l(k,d)$
    return $\nabla_{\hat{p}(\cdot|x)} f_\lambda(\hat{p}(\cdot|x)) := -\frac{1}{\lambda}[l(\cdot,\hat{y}) - l(\cdot,y_\lambda)]$
```

Mapping algorithm 1 to our cases, we receive the successive algorithms 2, 3, where one can see that the algorithms return the same values, proving that the two cases we proposed are equal. The $\lambda \in \mathbb{R}$ is a hyperparameter that controls the trade-off between the amount of information from the gradient and the original function.

In the end, we figured out one can obtain the gradient computed by the algorithm simply by applying the automatic differentiation to the function that will be referred

Algorithm 3: The algorithm with the mapping in case 2.

```
Function Forward($\hat{w} := \sum_k \hat{p}(k|x)l(k,\cdot)$):
    $\hat{y} := \arg\min_d \sum_k \hat{p}(k|x)l(k,\cdot)\,\text{one\_hot}_k(d)$
    save $\hat{w}$ and $\hat{y}$ for backward pass
    return $\hat{y}$
Function Backward($k^*$, $l$, $\lambda$):
    load $\hat{w}$ and $\hat{y}$ from forward pass
    $\hat{w}' := \hat{w} + \lambda \cdot l(k^*,\cdot)$
    $y_\lambda := \arg\min_d \sum_k (\hat{p}(k|x)l(k,\cdot) + \lambda \cdot l(k^*,\cdot))\,\text{one\_hot}_k(d)$
    return
      $\nabla_{\hat{p}(\cdot|x)} f_\lambda(\hat{p}(\cdot|x)) := -\frac{1}{\lambda}[\text{one\_hot}(\hat{y})^T l - \text{one\_hot}(y_\lambda)^T l] = -\frac{1}{\lambda}[l(\cdot,\hat{y}) - l(\cdot,y_\lambda)]$
```

to as the *Direct Loss*:

$$L^{\text{Direct}} = -\frac{1}{\lambda}\sum_{i=1}\Big(\min_d\big[\sum_k \hat{p}(k|x_i)l(k,d)\big] - \min_d\big[\sum_k \hat{p}(k|x_i)l(k,d) + \lambda l(k^*,d)\big]\Big), \tag{3.7}$$

where $\mp$ is paired with $\pm$. It is proven in Theorem 1 of (Song et al., 2016) that for $\lambda \to 0$ the gradient of this expression approaches the gradient of teh true risk.

Vlastelica et al. (2019) were encouraging for the use of a $\lambda > 0$, however we propose to use $\lambda < 0$, as we show that for all such λ, Direct Loss becomes an upper bound of the original loss function, which fits into the loss function minimization framework.

3.3.1 Upper/Lower bound proofs for different λ

We adopt similar notations from (Vlastelica et al., 2019) for the proof only. Let $w \in W \subseteq \mathbb{R}^N$ and $y \in Y$ a finite set of labels. Also consider a linear solver $\boldsymbol{c}(w,y) = w^T\phi(y)$, where $\phi(y)$ is a one-hot encoding of class label y. Define a solver as a function that maps

$$w \to y(w) \text{ such that } y(w) = \arg\min_{y\in Y} \boldsymbol{c}(w,y) \tag{3.8}$$

Consider $f(y)$ to be our piecewise constant function. Then define

$$\begin{aligned} f_\lambda(w) &= f(y_\lambda(w)) - \frac{1}{\lambda}\big[\boldsymbol{c}(w,y(w)) - \boldsymbol{c}(w,y_\lambda(w))\big] \\ &= -\frac{1}{\lambda}\big[\boldsymbol{c}(w,y(w)) - \boldsymbol{c}(w,y_\lambda(w)) - \lambda f(y_\lambda(w))\big] \end{aligned} \tag{3.9}$$

where

$$y_\lambda(w)) = \arg\min_{y \in Y}\{\boldsymbol{c}(w,y) + \lambda f(y)\}. \tag{3.10}$$

Note that equation (3.7) is a special case of (3.9).

Lemma 1. $\boldsymbol{c}(w,y) \geq \boldsymbol{c}(w,y(w))$ *for every* $w \in W$, $y \in Y$.

Proof. Holds trivially from the definition of $y(w)$, as we define it to be the minimum. □

Similarly,

Lemma 2. $\boldsymbol{c}(w,y) + \lambda f(y) \geq \boldsymbol{c}(w,y_\lambda(w)) + \lambda f(y_\lambda(w))$ *for every* $w \in W$, $y \in Y$ *and* $\lambda \in \mathbb{R}$.

Proof. Also hold trivially, this time from the definition of $y_\lambda(w)$. □

Theorem 1. *For all* $w \in W$, *if* $\lambda > 0$, *then*

$$f_\lambda(w) \leq f(y(w)) \tag{3.11}$$

and if $\lambda < 0$, *then*

$$f_\lambda(w) \geq f(y(w)). \tag{3.12}$$

Proof. We only prove without the loss of generality (3.12) as the proofs are similar. Let $\lambda < 0$. Using Lemma (2) we have for every $w \in W$, $y \in Y$

$$\begin{aligned}
\boldsymbol{c}(w,y) + \lambda f(y) &\geq \boldsymbol{c}(w,y_\lambda(w)) + \lambda f(y_\lambda(w)) \\
\boldsymbol{c}(w,y(w)) + \lambda f(y(w)) &\geq \boldsymbol{c}(w,y_\lambda(w)) + \lambda f(y_\lambda(w)) \\
&\qquad \text{(substitute y for y(w))} \\
\boldsymbol{c}(w,y(w)) - \boldsymbol{c}(w,y_\lambda(w)) + \lambda f(y(w)) &\geq \lambda f(y_\lambda(w)) \\
\frac{1}{\lambda}\big[\boldsymbol{c}(w,y(w)) - \boldsymbol{c}(w,y_\lambda(w))\big] + f(y(w)) &\leq f(y_\lambda(w)) \qquad \text{(divide by } \lambda < 0) \\
\frac{1}{\lambda}\big[\boldsymbol{c}(w,y(w)) - \boldsymbol{c}(w,y_\lambda(w))\big] + f(y(w)) &\leq f_\lambda(w)) + \frac{1}{\lambda}\big[\boldsymbol{c}(w,y(w)) - \boldsymbol{c}(w,y_\lambda(w))\big] \\
&\qquad \text{(substitute (3.9))} \\
f(y(w)) &\leq f_\lambda(w).
\end{aligned}$$

□

We have proven that the piecewise constant function f is less or equal than f_λ for all $\lambda < 0$, which is what we wanted to prove. Therefore, using (3.7) with a negative lambda, we upper bound the empirical risk. However, for very small $|\lambda|$ values, the sizes of the margins between the original function and Direct Loss are small, still allowing for flat regions with zero gradient. We tackle this by using the smooth minimum function instead of a plain minimum, i.e. $min_\beta(x) = -\frac{1}{\beta}\log\sum_k e^{-\beta x_k}$, where β is the smoothing parameter. This breaks the upper/lower bound guarantees but we hypothesize that it is not detrimental for learning. This is visualized later in figures 4.e and 4.f.

3.4 Integral Loss method

It may be that one would want to calibrate with respect to a set of loss functions, rather than a single loss function like in definition 1. Let us redefine true decision calibration as a more general case:

Definition 2. *Let $r : \Delta \to \Delta$ be the distribution calibration mapping and $\hat{p}$ the probability predictor. Consider $\mathcal{L}$ to a be set of loss matrices. The predictor $\hat{p}$ is considered to be calibrated if the true risk of the predictor $\hat{p}$ is equal to the true risk of the recalibrated predictor $r(\hat{p}(\cdot|x))$ for all losses from the set $\mathcal{L}$. This reads as*

$$\mathbb{E}_X\mathbb{E}_{Y\sim p^*(\cdot|X)}[l(Y, q(\hat{p}(\cdot|X)))] = \mathbb{E}_X\mathbb{E}_{Y\sim p^*(\cdot|X)}[l(Y, q(r(\hat{p}(\cdot|X))))],\ \forall l \in \mathcal{L}. \tag{3.13}$$

To this end, we focus on a specialized case of true decision calibration for sets $\mathcal{L} = \{(1-\epsilon)l_0 + \epsilon l_1 \mid \forall \epsilon \in [0,1]\}$, where l_0 and l_1 are arbitrary loss matrices. In other words, we specialize on sets $\mathcal{L}$ that are made from all loss matrices in the convex hull of two loss matrices l_0 and l_1. One clear application of this would be decision making with a reject option, for which we do not know what the real reject loss is. Another application could be the case, where we know that a certain prediction may cost us a value in an interval.

Similarly to the case in definition 1, given a single loss $l \in \mathcal{L}$, the difference between the left-hand side and the right-hand side is a measure of miscalibration. The integral over all losses $l \in \mathcal{L}$ is also a measure of miscalibration. Thus, minimizing this integral miscalibration is equivalent to minimizing the integrated Empirical Risk over an invertible calibration mapping.

For that, we propose another loss criterion, whose inputs are two loss matrices and the loss criterion tries to optimize to perform well on average for a convex combination of the two loss matrices. Therefore, for optimizing the calibration parameters, we would like to optimize the following:

$$\min_\theta \int_{\epsilon=0}^{1} \frac{1}{n} \sum_{i=1}^{n} l_\epsilon(y_i, \arg\min_{d\in\mathcal{A}} \sum_k \hat{p}_\theta(k|x_i) l_\epsilon(k,d))\, d\epsilon =$$
$$\min_\theta \frac{1}{n} \sum_{i=1}^{n} \int_{\epsilon=0}^{1} l_\epsilon(y_i, \arg\min_{d\in\mathcal{A}} \sum_k \hat{p}_\theta(k|x_i) l_\epsilon(k,d))\, d\epsilon, \tag{3.14}$$

where the sum and the integral can be interchanged by Tonelli's theorem as we assume $l_\epsilon \geq 0$. The θ are the calibration parameters and $l_\epsilon = l_0 + \epsilon l_1$, $\epsilon \in [0,1]$. However, in this form it is not clear that we can use the minimization framework via gradient descent. We will show that the integral can be computed analytically, resulting in a differentiable function over $\hat{p}_\theta$, hence over θ.

3.4.1 Integral Loss method for 0/1 loss

Let us start by deriving the method for a 0/1 loss matrix with ϵ reject loss, e.g. for $|\mathcal{Y}| = 2$ and $|\mathcal{A}| = |\mathcal{Y}| + 1 = 3$:

$$l_\epsilon = \begin{pmatrix} 0 & 1 & \epsilon \\ 1 & 0 & \epsilon \end{pmatrix}$$

The number of actions is the number of classes with an addition of the reject option. We refer to the reject option as d_ϵ. Clearly, this matrix can be decomposed into two matrices l_0 and l_1 such that $l_\epsilon = l_0 + \epsilon l_1$ holds.

Firstly, the term $\arg\min_d \sum_k \hat{p}_\theta(k|x) l_\epsilon(k,d)$ can be simplified. Under the given 0/1 loss function, notice that for every decision d and label k, it holds that:

$$\hat{p}_\theta(k|x) l_\epsilon(k,d) = \begin{cases} \hat{p}_\theta(k|x)\epsilon & d = d_\epsilon \\ 0 & d = k \\ \hat{p}_\theta(k|x) & \text{otherwise.} \end{cases}$$

From here, one can see that for every decision d, the sum can be factorized accordingly:

$$\sum_k \hat{p}_\theta(k|x) l_\epsilon(k,d) = \begin{cases} \epsilon & d = d_\epsilon \\ 1 - \hat{p}_\theta(d|x) & \text{otherwise.} \end{cases}$$

In other words, we can simplify the term into

$$\arg\min(1 - \hat{p}(0|x), 1 - \hat{p}(1|x), \ldots, 1 - \hat{p}(\mathcal{Y}|x), \epsilon).$$

This turns down into comparing $\delta := 1 + \min_{k\in\mathcal{Y}} -\hat{p}(k|x) = 1 - \max_k \hat{p}_\theta(k|x_i)$ with ϵ. If $\epsilon < \delta$, then we should choose the reject option, since the loss for rejecting is lower. Otherwise we choose the class with the highest probability.

$$\int_0^1 l_\epsilon(y, \arg\min_{d\in\mathcal{A}} \sum_k \hat{p}_\theta(k|x) l_\epsilon(k,d))\, d\epsilon = \tag{3.15}$$

$$\int_0^\delta l_\epsilon(y,\epsilon)\, d\epsilon + \int_\delta^1 l_\epsilon(y, \arg\max_{k\in\mathcal{Y}} \hat{p}_\theta(k|x))\, d\epsilon = \tag{3.16}$$

$$\left[\frac{\epsilon^2}{2}\right]_0^\delta + [\![y \neq \arg\max_{k\in\mathcal{Y}} \hat{p}_\theta(k|x)]\!] \Big[\epsilon\Big]_\delta^1 = \tag{3.17}$$

$$\frac{(1-\max_k \hat{p}_\theta(k|x))^2}{2} + [\![y \neq \arg\max_{k\in\mathcal{Y}} \hat{p}_\theta(k|x)]\!] \max_k \hat{p}_\theta(k|x). \tag{3.18}$$

The set where this is not differentiable in θ is of measure zero, as there are single points that are not differentiable due to the maximum operators and Iverson brackets.

3.4.2 Integral Loss method for any loss

We now compute Integral Loss for a general $\mathcal{L} = conv(l_0, l_1)$. Restricting the loss to the 0/1 loss allowed us to derive a nicer form of (3.14) using some simple properties of the 0/1 loss. Having no assumptions other than non-negativity about the loss matrix, we have to use other techniques.

Firstly, let $l_\epsilon = l_0 + \epsilon l_1$ for any given l_0, l_1 and $R_\epsilon(d) = \sum_k \hat{p}(k|x) l_\epsilon(k,d)$, for a given decision d. Then it holds that

$$\begin{aligned}
&\min_\theta \frac{1}{n}\sum_{i=1}^n \int_0^1 l_\epsilon(y_i, \arg\min_{d\in\mathcal{A}} \sum_k \hat{p}_\theta(k|x_i) l_\epsilon(k,d))\, d\epsilon = \\
&\min_\theta \frac{1}{n}\sum_{i=1}^n \int_0^1 l_\epsilon(y_i, \arg\min_{d\in\mathcal{A}} R_\epsilon(d))\, d\epsilon = \\
&\min_\theta \frac{1}{n}\sum_{i=1}^n \sum_{d\in\mathcal{A}} \int_0^1 l_\epsilon(y_i,d) [\![R_\epsilon(d) \leq R_\epsilon(d')\ \forall d' \in \mathcal{A}]\!]\, d\epsilon = \\
&\min_\theta \frac{1}{n}\sum_{i=1}^n \sum_{d\in\mathcal{A}} \int_0^1 \big(l_0(y_i,d) + \epsilon l_1(y_i,d)\big) [\![R_\epsilon(d) \leq R_\epsilon(d')\ \forall d' \in \mathcal{A}]\!]\, d\epsilon,
\end{aligned} \tag{3.19}$$

in other words, we sum over all decisions, but the Iverson brackets filter out only the decision that is for the given ϵ the minimum. The integral is linear in ϵ, therefore we can factor out l_0 and l_1. What is left is to analyze the Iverson brackets. Denote $\xi(d)$ the set of all ϵ for a given decision d, where the Iverson brackets are true, i.e.

$$\xi(d) = \{\epsilon \in [0,1] \mid R_\epsilon(d) \leq R_\epsilon(d'),\ \forall d' \in \mathcal{A}\}. \tag{3.20}$$

Having this, we can now rewrite (3.19) using the $\xi(d)$ notation as:

$$\min_\theta \frac{1}{n} \sum_{i=1}^{n} \sum_{d\in\mathcal{A}} \big(l_0(y_i, d) \int_{\epsilon\in\xi(d)} 1\, d\epsilon + l_1(y_i, d) \int_{\epsilon\in\xi(d)} \epsilon\, d\epsilon\big). \tag{3.21}$$

Let us further analyze the $\xi(d)$ set. For a given decision d, the set $\xi(d)$ will contain all $\epsilon \in [0,1]$ that satisfy all of the $|\mathcal{A}|$ inequalities. Further, these inequalities will be satisfied for only certain ϵ – the intersection of ϵ for which all the inequalities are satisfied, giving us the bounds for the integrals. We dive deeper into the inequalities in the following way:

$$\begin{aligned}
&R_\epsilon(d) \leq R_\epsilon(d'),\ \forall d' \in \mathcal{A} \\
&R_\epsilon(d) - R_\epsilon(d') \leq 0,\ \forall d' \in \mathcal{A} \\
&\sum_k \hat{p}(k|x) l_\epsilon(k,d) - \sum_k \hat{p}(k|x) l_\epsilon(k,d') \leq 0,\ \forall d' \in \mathcal{A} \qquad \text{(definition of } R_\epsilon(d)\text{)} \\
&\sum_k \hat{p}(k|x)\big(l_\epsilon(k,d) - l_\epsilon(k,d')\big) \leq 0,\ \forall d' \in \mathcal{A} \\
&\sum_k \hat{p}(k|x)\big(l_0(k,d) + \epsilon l_1(k,d) - l_0(k,d') - \epsilon l_1(k,d')\big) \leq 0,\ \forall d' \in \mathcal{A} \\
&\sum_k \hat{p}(k|x)(l_0(k,d) - l_0(k,d')) + \epsilon \sum_k \hat{p}(k|x)(l_1(k,d) - l_1(k,d')) \leq 0,\ \forall d' \in \mathcal{A}.
\end{aligned} \tag{3.22}$$

Denote

$$a_{d'}(d) := \sum_k \hat{p}(k|x)(l_0(k,d) - l_0(k,d')) \tag{3.23}$$

$$b_{d'}(d) := \sum_k \hat{p}(k|x)\epsilon(l_1(k,d) - l_1(k,d')), \tag{3.24}$$

then we need to solve

$$a_{d'}(d) + \epsilon b_{d'}(d) \leq 0,\ \forall d' \in \mathcal{A}. \tag{3.25}$$

From here, we solve for ϵ based on the values of $a_{d'}(d), b_{d'}(d)$. Denote $D^-(d)$, $D^+(d)$, $D^0(d)$ the sets of d', based on the sign of $b_{d'}(d)$, that is

$$D^+(d) = \{d' \mid b_{d'}(d) > 0\} \tag{3.26}$$

$$D^-(d) = \{d' \mid b_{d'}(d) < 0\} \tag{3.27}$$

$$D^0(d) = \{d' \mid b_{d'}(d) = 0\}. \tag{3.28}$$

If $b_{d'}(d) > 0$, then $\epsilon \leq -\frac{a_{d'}(d)}{b_{d'}(d)}$. This shows the upper bound (UB) of the integration range. We want the ϵ that satisfies this inequality for all $d' \in \mathcal{A}$. Similarly, we will want this for the case where $b_{d'}(d) < 0$, which will give us the lower bound (LB).

$$\mathrm{UB}(d) = \min_{d' \in D^+} -\frac{a_{d'}(d)}{b_{d'}(d)} \tag{3.29}$$

$$\mathrm{LB}(d) = \max_{d' \in D^-} -\frac{a_{d'}(d)}{b_{d'}(d)} \tag{3.30}$$

Further, we clamp UB and LB between 0 and 1 to satisfy the $\epsilon \in [0, 1]$ constraint:

$$\mathrm{UB}(d) = \min(\max(\min_{d' \in D^+} -\frac{a_{d'}(d)}{b_{d'}(d)}, 0), 1) \tag{3.31}$$

$$\mathrm{LB}(d) = \min(\max(\max_{d' \in D^-} -\frac{a_{d'}(d)}{b_{d'}(d)}, 0), 1). \tag{3.32}$$

As for the case where $b_{d'}(d) = 0$, if there exists a $d' \in D^0(d)$ such that $a_{d'}(d) > 0$, then the (3.25) is never satisfied and therefore $R_\epsilon(d)$ will not be the minimum solution, i.e.

$$[\![valid(d)]\!] := (1 - [\![\exists d' : b_{d'}(d) = 0 \ \wedge \ a_{d'}(d) > 0]\!]). \tag{3.33}$$

Substituting it all back into (3.21), we get

$$\begin{aligned}
&\min_\theta \frac{1}{n} \sum_{i=1}^{n} \sum_{d \in \mathcal{A}} \big(l_0(y_i, d) \int_{\epsilon \in \xi(d)} 1 \, d\epsilon + l_1(y_i, d) \int_{\epsilon \in \xi(d)} \epsilon \, d\epsilon \big) = \\
&\min_\theta \frac{1}{n} \sum_{i=1}^{n} \sum_{d \in \mathcal{A}} [\![valid(d)]\!] \Big(l_0(y_i, d) \, [\epsilon]_{\mathrm{LB}(d)}^{\mathrm{UB}(d)} + l_1(y_i, d) \left[\frac{\epsilon^2}{2} \right]_{\mathrm{LB}(d)}^{\mathrm{UB}(d)} \Big).
\end{aligned} \tag{3.34}$$

For this, the non-differentiable set by θ is of measure zero. Differentiating by θ will be involved in every conditional probability, which is in every part, where $a_{d'}(d)$ and

$b_{d'}(d)$ is – the Iverson bracket and the lower/upper bounds. The Iverson bracket is a stepwise characteristic function, having only single non-differentiable points, which are of measure zero. In the lower/upper bound calculations, min and max operators are also non-differentiable in single points. Lastly, $a_{d'}(d)$ and $b_{d'}(d)$ are differentiable as they are simply convex combinations over the possible classes.

The complexity for one pair (x, y) is $\mathcal{O}(|\mathcal{Y}||\mathcal{A}|)$, since we can multiply the probabilities with all decisions d and d' and then perform the subtraction, resulting in a complexity of $2|\mathcal{Y}||\mathcal{A}| \in \mathcal{O}(|\mathcal{Y}||\mathcal{A}|)$. The naive implementation will lead to a complexity of $\mathcal{O}(|\mathcal{Y}||\mathcal{A}|^2)$.

Comparing Integral Loss to Direct Loss, although we are not capable to optimize for a specific loss matrix, we got rid of two unclear hyperparameters, λ and β specifically. For them, we do not know the interval from which to sample for the best results, and thus we have to opt for timely grid searches.

The function is later visualized and explained in figures 4.e and 4.f.

3.5 Calibration for prior shift

Using our criterions, we can also calibrate for prior shift. Assume we have an estimate of the prior probability for the target distribution $P_{test}(y)$. The task is to estimate the new conditional probability $P_{test}(y|x)$ given the test prior probability. For prior shift, we assume that $P_{test}(x|y) = P_{train}(x|y)$ as described in 2.7.2. From the Bayes Theorem we know that

$$P(x|y) = \frac{P(y|x)P(x)}{P(y)},$$

which for different train and test distributions and the prior shift assumption gives us

$$\frac{P_{test}(y|x)P_{test}(x)}{P_{test}(y)} = \frac{P_{train}(y|x)P_{train}(x)}{P_{train}(y)}. \tag{3.35}$$

From there, solving for $P_{test}(y|x)$, we get

$$P_{test}(y|x) = P_{train}(y|x)\frac{P_{test}(y)}{P_{train}(y)}\frac{P_{train}(x)}{P_{test}(x)} \propto P_{train}(y|x)\frac{P_{test}(y)}{P_{train}(y)}, \tag{3.36}$$

which will be also used for making decisions under the test prior probabilities. Given samples from the held-out training distribution, we can estimate the expectation of any function f with respect to the test distribution by adding the importance weights:

$$\sum_{x,y} P_{test}(x,y)f(x,y) = \sum_{x,y} P_{train}(x,y)\frac{P_{test}(y)}{P_{train}(y)}f(x,y) = \mathbb{E}_{P_{train}(x,y)}\Big[\frac{P_{test}(y)}{P_{train}(y)}f(x,y)\Big].$$

The true risk $R(q)$ thus takes the form

$$R(q) = \mathbb{E}_{P_{train}(x,y)}\Big[\frac{P_{test}(y)}{P_{train}(y)}l(y,q(x))\Big]. \tag{3.37}$$

For completeness, the model risk $\hat{R}(q)$ can also be estimated as

$$\begin{aligned}\hat{R}(q) &= \mathbb{E}_{P_{test}(x)}\Big[\sum_y \hat{P}_{test}(y|x)l(y,q(x))\Big]\\ &= \sum_x P_{test}(x)\Big[\sum_y \hat{P}_{train}(y|x)\frac{P_{test}(y)}{P_{train}(y)}\frac{P_{train}(x)}{P_{test}(x)}l(y,q(x))\Big]\\ &= \sum_x P_{train}(x)\Big[\sum_y \hat{P}_{train}(y|x)\frac{P_{test}(y)}{P_{train}(y)}l(y,q(x))\Big]\\ &= \mathbb{E}_{P_{train}(x)}\Big[\sum_y \hat{P}_{train}(y|x)\frac{P_{test}(y)}{P_{train}(y)}l(y,q(x))\Big]. \end{aligned} \tag{3.38}$$

For our experiments, we will use the equation (3.36) to get the decisions enhanced by the importance weights on the test set and then use the equation (3.37) to compute the empirical risk and asses the performance under the prior shift, without having to resample the dataset according to the new test set distribution.

3.5.1 Using Integral Loss for prior shift

Having shown that in the prior shift case the risks can be easily estimated using importance sampling, we can create a realistic scenario for which the Integral Loss could be of use.

Let $P'(y) \neq P_{train}(y)$ be another prior distribution. It could happen that the test prior distribution $P_{test}(y)$ is a convex combination of the two, i.e.

$$P_{test}(y) = (1-\epsilon)P_{train}(y) + \epsilon P'(y), \tag{3.39}$$

where $\epsilon \in [0,1]$, unknown during calibration time. Thus, we would like to calibrate the probabilities so that it does well on average for any ϵ during test time which is what the Integral Loss was designed for. Reminding that the basic form of Integral Loss is:

$$\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} \int_{0}^{1} l_{\epsilon}(y_i, \arg\min_{d \in \mathcal{A}} \sum_{k} \hat{p}_{\theta}(k|x_i) l_{\epsilon}(k, d)) \, d\epsilon,$$

in which we can substitute

$$l_{\epsilon}(k, d) := \frac{P_{test}(k)}{P_{train}(k)} l(k, d) = \frac{(1-\epsilon)P_{train}(k) + \epsilon P'(k)}{P_{train}(k)} l(k, d).$$

From here, it is trivial to rewrite the equation in the $l_0 + \epsilon l_1$ form to fit the Integral Loss framework:

$$\begin{aligned} l_{\epsilon}(k, d) &= \frac{(1-\epsilon)P_{train}(k) + \epsilon P'(k)}{P_{train}(k)} l(k, d) \\ l_{\epsilon}(k, d) &= \frac{P_{train}(k) + \epsilon(P'(k) - P_{train}(k))}{P_{train}(k)} l(k, d), \end{aligned}$$

where one can see that:

$$l_0(k, d) = \frac{P_{train}(k)}{P_{train}(k)} l(k, d) = l(k, d) \tag{3.40}$$

$$l_1(k, d) = \frac{P'(k) - P_{train}(k)}{P_{train}(k)} l(k, d). \tag{3.41}$$

3.6 Experimental workflow of the calibration process

For our experimental purposes, we employ a slightly different process. Firstly, we use nested cross validation for each calibration method. That is to choose the correct hyperparameters first and then to evaluate the calibration method with a confidence level. We show this overview in figure 3.a.

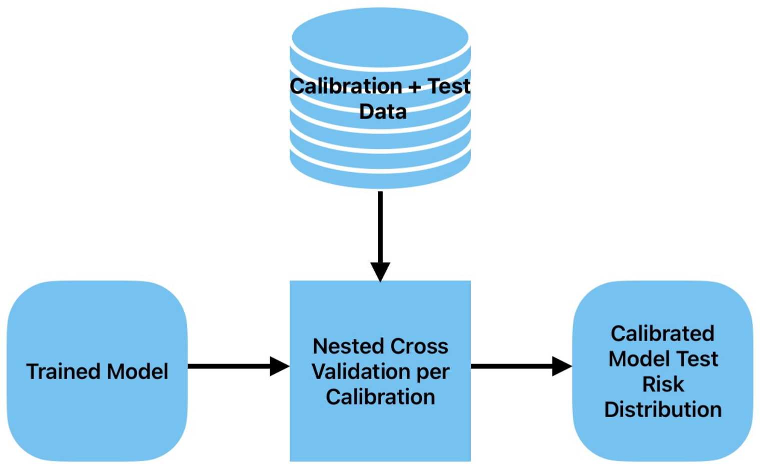
Calibration + Test Data
Trained Model
Nested Cross Validation per Calibration
Calibrated Model Test Risk Distribution

Figure 3.a: The overview of the workflow we used in our experiments. We performed nested cross validation for choosing the optimal hyperparameters and for having confidence intervals for our results.

To explain nested cross validation in detail, we visualize the process in figure 3.b. To mantain class balance, we use stratified k-fold cross validation. The hyperparameters are chosen through grid search and the best performing hyperparameters are then used to fit the calibration method and calibrate the model. The hyperparameters that we need to choose are

- λ - the linearization parameter of Direct Loss,
- β - the smoothed minimum parameter of Direct Loss,
- learning rate - the learning rate for training the calibration method through gradient descent.

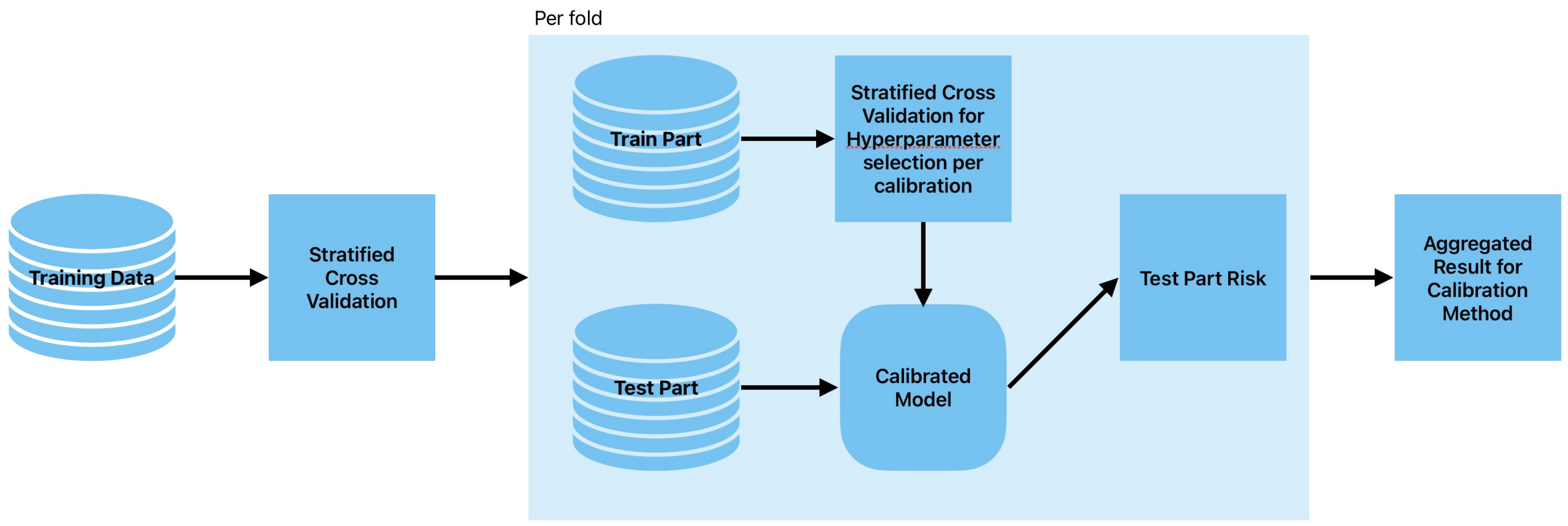
Training Data
Stratified Cross Validation
Per fold
Train Part
Test Part
Stratified Cross Validation for Hyperparameter selection per calibration
Calibrated Model
Test Part Risk
Aggregated Result for Calibration Method

Figure 3.b: The nested cross validation process in details. For every fold in the stratified cross validation, we perform hyper parameter optimization using grid search and afterwards fit calibration method using the selected hyperparameters and evaluate the calibration on the test fold.

Chapter 4

Experiments

This chapter will show some basic analyses and the performance results of our developed loss criterions. We start with explaining the technical setup that we used, show how the loss criterions behave in simple one-dimensional experiments and, lastly, show the performance of the loss criterions on three different situations, where:

- the loss matrix is known exactly,
- the loss matrix is known as a convex combination of two known matrices,
- the prior shift adaptation for prior distribution is represented as a convex combination.

In the end of this chapter we provide a discussion of the results.

4.1 Technical setup

To first get a model to calibrate, we either train our own neural networks or take already trained networks for the given dataset. We chose one artificial dataset and then two realistic datasets.

4.1.1 CIFAR–100

First, we use a standard image classification dataset, CIFAR–100 (Krizhevsky et al., 2009), but in the superclass version. The superclass version contains 20 classes, allowing us to create artificial loss matrices with ease. The dataset is split into 40000 training images, 10000 calibration images and 10000 test images. We use ResNet32 (He et al., 2016) for the classification task, with a 76% accuracy on the test set. Although we know that there are pretrained models with accuracy 90% and more, we intentionally keep this model. For this data we wanted to test how the model's Empirical Risk improves with calibration to a certain loss and if we had used too good of a model, the improvement might not be visible easily. The loss matrices we use are in the form of:

$$l(y, \hat{y}) = \begin{cases} 0 & y = \hat{y}, \\ X & y \neq \hat{y} \text{ and } y \text{ is a dangerous class}, \\ 1 & \text{otherwise}, \end{cases}$$

where X is chosen to be a high value to simulate the loss of making a wrong decision during safety-critical applications. We choose the dangerous classes to be 18 and 19, which are vehicle superclasses.

4.1.2 Danish Fungi

The Danish Fungi (Picek et al., 2022) is a fine-grained dataset with a long-tailed distribution of prior probabilities of the classes. The authors also provide a pretrained model which we used. We hand annotated 167 of the classes based on their edibility – deadly poisonous, poisonous, inedible, edible bad, edible, edible good. We normalized the predicted probabilities so that the output is the probability of the edibility class. The remapped class distribution is visualized in figure 4.a. Here, the objective is to then decide whether or not should a cook buy the given mushroom. With this in mind, we designed the loss matrix as:

	Accept	Reject
deadly_poisonous	10000	0
poisonous	1000	0
inedible	100	0
edible_bad	40	0
edible	0	10
edible_good	0	20

In this new classification task we achieved a 90% accuracy. We provide some data examples in figure 4.b. For more information regarding the dataset we refer the reader to the main paper (Picek et al., 2022).

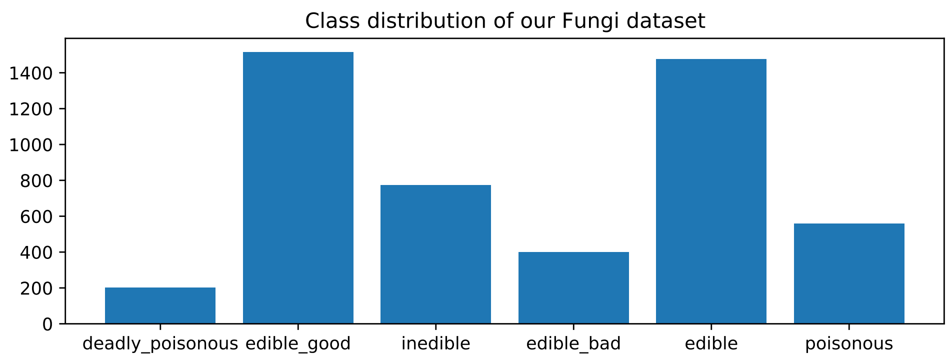

Figure 4.a: The distribution of edibility classes in the remapped Fungi validation + test set.

Figure 4.b: Data examples of the Danish Fungi dataset.

4.1.3 Ham10000

The Ham10000 dataset (Tschandl et al., 2018) is a cancer skin lesion image dataset, examples of the dataset shown in figure 4.c. Using this dataset we design the experiment to decide whether the patient should be given a treatment or not based on the prediction of the classifier. The training was performed on 75% of the data, the rest was used for calibrating (15%) and testing (10%). We design the loss matrix similarly as in (Zhao et al., 2021), with the added normalization to 0:

	no treat	treat
akiec	11	0
bcc	11	0
bkl	0	4
df	0	1.5
nv	0	2.5
mel	9.5	0
vasc	0	2

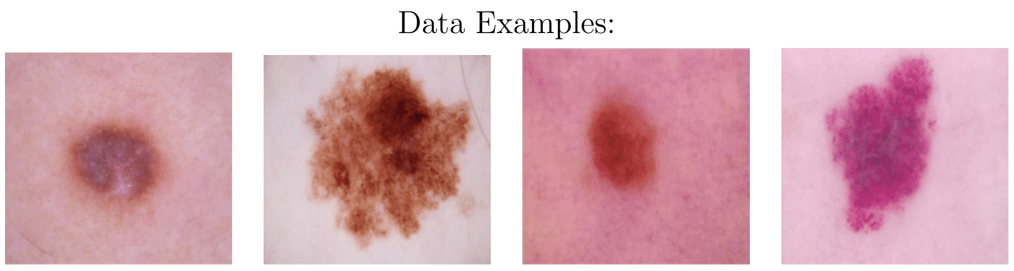

Figure 4.c: Data examples of the Ham10000 dataset.

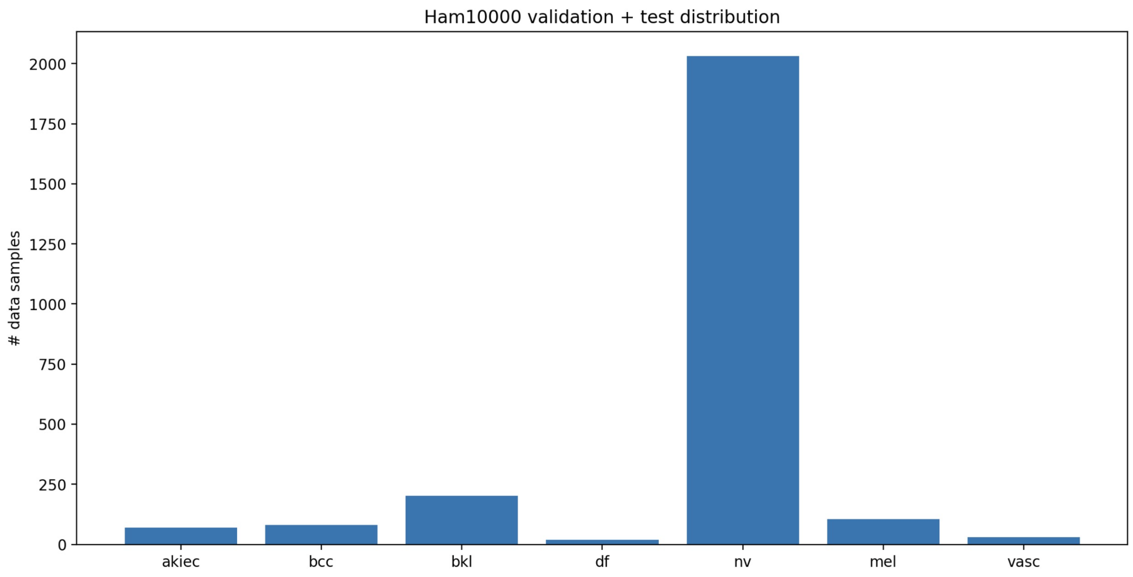

Figure 4.d: The distribution of Ham10000 classes in the validation and test set.

For one experiment we also add a reject option as an "*I do not know*" decision with a constant loss for every class to allow for uncertainty for the model. We used DenseNet121 (Huang et al., 2017) achieving 90% accuracy on the validation data. The data distribution of the test and validation set can be seen on figure 4.d.

4.2 Pilot experiments of the loss criterions

We first show the criterions with different parameters on the one-dimensional probability simplex. Consider a binary classification problem, where the true class is 1, a 0/1 loss matrix and no reject option. In this problem, we visualize Direct Loss with $\lambda \in -1, -0.1$ and no β min smoothing to show the effect of linearization. As for Integral Loss, we have to opt for a different problem setting, as Integral Loss cannot be optimized with respect to a single loss matrix. The l_0 matrix is the 0/1 loss matrix with a reject option that costs 0 and the l_1 matrix is a null matrix with a reject option that costs 1. For comparisons, we also include NLL and the Empirical Risk function. This all is shown in figure 4.e.

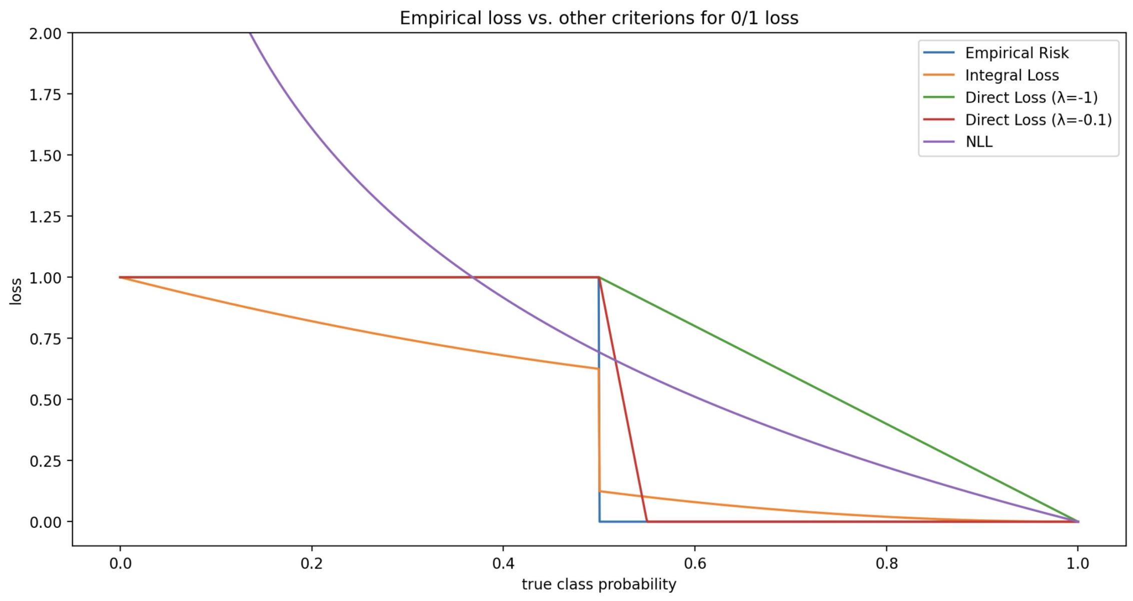

Figure 4.e: The loss criterions visualized on a one-dimensional probability simplex with a 0/1 loss and the true class being 1. Note, that using a negative λ leads to an upper bound of the Empirical Risk.

In the figure, empirical risk shows loss 1 for all probabilities that are less than 0.5, and 0 otherwise. Direct Loss with $\lambda = -1$ shows how the stepwise functions is linearized, however, the gradients in the probabilities in the interval $[0, 0.5]$ are 0. Direct Loss with $\lambda = -0.1$ shows a steeper linearization, imitating the Empirical Risk function better, but contains a wider space, where the gradients are uninformative. Integral Loss has a quadratic curve on both sides of 0.5.

Further, we show the effect of TS on the CIFAR–100 with superclasses dataset with different optimization criterions. The loss matrix is designed in the following way:

$$l(y, \hat{y}) = \begin{cases} 0 & y = \hat{y}, \\ 10000 & y \neq \hat{y} \;\&\; y \in \{18, 19\} \\ 1 & \text{otherwise.} \end{cases}$$

For Integral Loss, we design the l_0 and l_1 matrices so that $l_0 + 0.5l_1$ will sum into the same loss matrix l. Direct Loss is not β smoothed and λ was set to be $-\frac{1}{\max_{(y,d)\in\mathcal{Y}\times\mathcal{A}} l(y,d)} = -1e-05$. Further, we also show NLL for comparison and the Empirical Risk on the test set, to show how well the criterions generalize. The results are in figure 4.f.

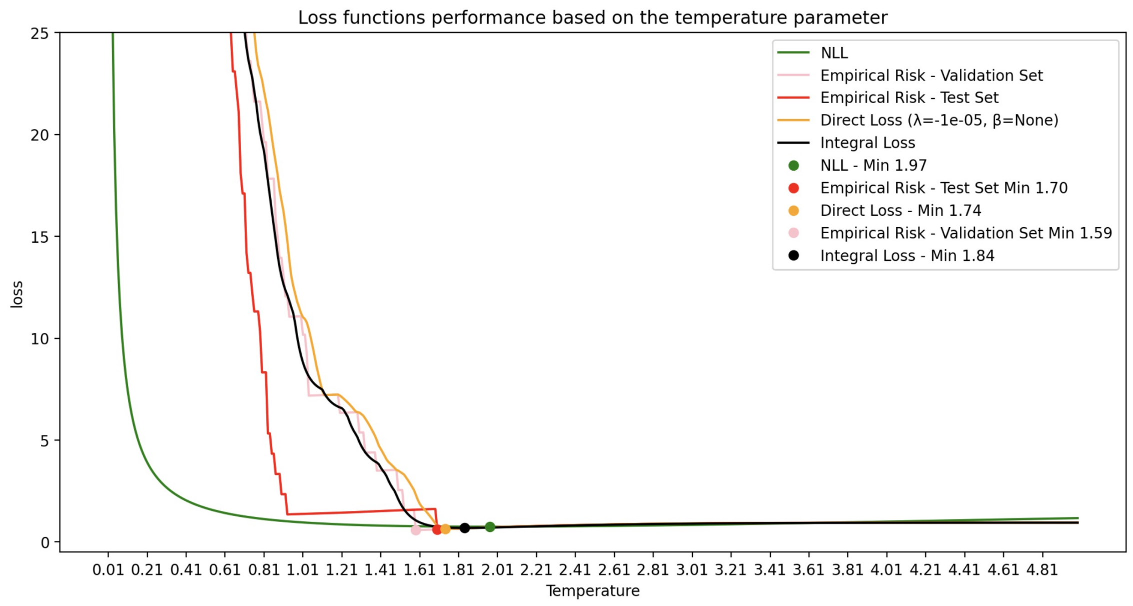

Figure 4.f: The loss criterions performing on the CIFAR–100 superclass set based on temperature scaling calibration. Note, that using a negative λ value gives us an upper bound of the Empirical Risk. We can see that in this experiment, the best temperature parameter is achieved by Direct Loss and then by Integral Loss. Optimizing based on the performance on the Empirical Risk we would overfit on the validation set and perform bad on the test set afterwards.

Here, if any loss criterion would perfectly fit the validation empirical risk, it would perform bad on the test set, as there is a misclassified example in the test set by the neural network. Having chosen good enough hyperparameters, Direct Loss accurately chooses the temperature so that it generalizes well to the test set as well. Integral loss is more conservative, and NLL is the most conservative. Note that NLL would be the same for every loss matrix. Furthermore, it can be seen that both Direct Loss and Integral Loss copy the Empirical Risk curve.

4.3 Loss matrix known exactly

In this section we describe the situation for which we know the exact loss matrix used during test time, thus we do not include Integral Loss. We perform the experiment on the Danish Fungi and the Ham10000 datasets. The loss matrices used are as described in the datasets section. We try the performance of TS, BCTS, VS with different criterions.

4.3.1 Danish Fungi

The results for this experiment are shown in the figure 4.g. In this experiment, our aggregation function is not the mean but the sum, i.e. the Empirical Risk multiplied by the number of samples to overview what losses from the loss matrix applied. Further, we create 15 folds in total to asses some confidence to our results.

In this experiment, no calibration performs terribly, BCTS calibration performs the best, followed by TS and VS lastly. Finally, Direct Loss is outperforming all other optimization criterions. One would also expect that VS would perform the best, as it is a generalization of both TS and BCTS, however it seems that it performs the worst. We hypothesize that it is because of overfitting.

Results

Parametrization	Calibration Criterion	Test Sum of Losses
No calibration	—	2117 ± 886
TS	NLL	944 ± 32
	ECE	944 ± 40
	Direct Loss	$\mathbf{758 \pm 35}$
BCTS	NLL	897 ± 31
	ECE	823 ± 34
	Direct Loss	$\mathbf{730 \pm 31}$
VS	NLL	1416 ± 649
	ECE	1095 ± 70
	Direct Loss	$\mathbf{779 \pm 68}$

Figure 4.g: The results of the different parametrizations and loss criterions on the dataset. Results were aggregated from 15 folds and are showing the mean and one standard deviation.

4.3.2 Ham10000

We perform these experiments similarly as we did with the Danish Fungi dataset. Aggregated as sum with the given loss matrix, created from 15 validation folds. For this dataset, we cannot find a clear best performing parametrization nor criterion, if we consider the standard deviations in. Thus, we further analyze the difference between the true risk and the model risk for different loss criterions and parametrizations, which we show in figures 4.i, 4.j and 4.k. Each of these figures differs by the parametrization parameter, where (a) shows the model risk distribution and the true/empirical risk distribution, (b) shows the statistical effect of calibration on the true risk (positive is improvement) and (c) shows the same effect as (b) but for NLL and Direct Loss.

We notice that the model risk underestimates the true risk, and the goal would be that the calibration brings these distribution closer together. For BCTS we see that some folds get farther away and for VS most of the folds get farther and therefore we hypothesize overfitting. On the other hand, TS puts the distributions closer together. In (b) and (c) we mostly notice that calibration has a positive effect for the decision making and there are not many differences in NLL and Direct Loss for this dataset.

Results

Parametrization	Calibration Criterion	Test Sum of Losses
No calibration	—	145.77 ± 4.57
TS	NLL	151.53 ± 4.21
	ECE	150.47 ± 4.08
	Direct Loss	150.73 ± 4.60
BCTS	NLL	146.77 ± 4.3
	ECE	144.53 ± 3.97
	Direct Loss	142.47 ± 4.71
VS	NLL	147.23 ± 5.24
	ECE	151.33 ± 3.18
	Direct Loss	146.77 ± 4.68

Figure 4.h: The calibration results for the Ham10000 dataset for different loss criterions and calibration methods. No statistically significant improvements.

4.4 Loss matrix known as a convex combination of two matrices during test time

To also assess the performance of Integral Loss, we design experiments for that. In this task, we know during calibration time that the loss matrix will be a convex combination of two matrices with a coefficient ϵ which will be told during test time after the calibration. More specifically, confidence calibration with a reject option is a convex combination of two matrices or the task where the missing the dangerous class has a high loss is a convex combination of two matrices. All the experiments are performed on 5 folds to asses a confidence in the results of all parametrizations

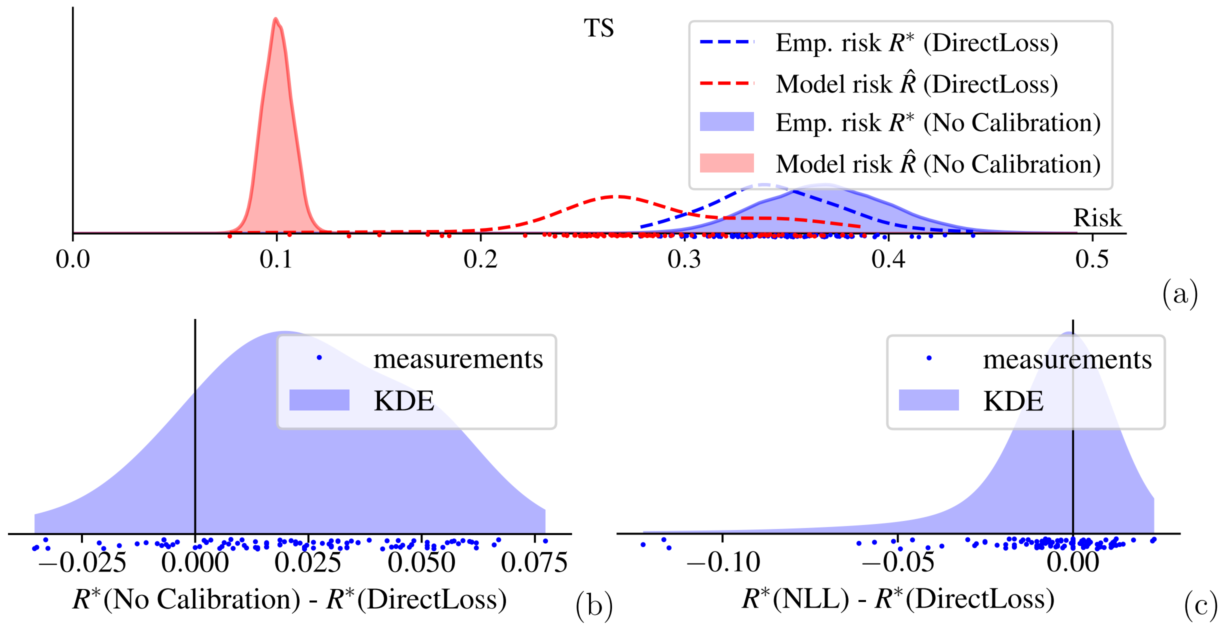

Figure 4.i: The performance of TS on Ham10000 dataset, showing the model and true risks before and after calibration. (a) The model risk and true risk compared with and without calibration. The gap is closer after calibration. (b) The effect of calibrating with Direct Loss and with no calibration in the true risk. More positive results mean an improvement, therefore here we see a positive result. (c) Effect of calibrating with Direct Loss and with NLL in the true risk. Similarly, positive result are an improvement, here we have are mainly distributed around 0.

and criterions and are visualized as a plot of the dependance of the Empirical Risk on the test set and the given ϵ. However, due to Direct Losses computational complexity in terms of hyperparameter tuning, we only computed Direct Loss for $\epsilon \in [0, 0.25, 0.5, 0.75, 1]$ (for Direct Loss known during calibration time!).

4.4.1 Confidence calibration on CIFAR–100 superclasses

For this experiment, we use the CIFAR–100 superclass version dataset with the 0/1 loss with a reject option with a loss in the interval $[0, 1]$ unknown during the calibration time. Integral Loss l_0 matrix is the 0/1 loss and l_1 matrix has the loss 1 only for the reject option, 0 otherwise. The graphs are shown in figure 4.l for every parametrization (TS, BCTS, VS).

The performance does not really differ given the parametrization and the loss criterion besides the ECE for BCTS and VS. We again believe the reason for this is overfitting with the larger number of parameters. Further, Direct Loss seems to

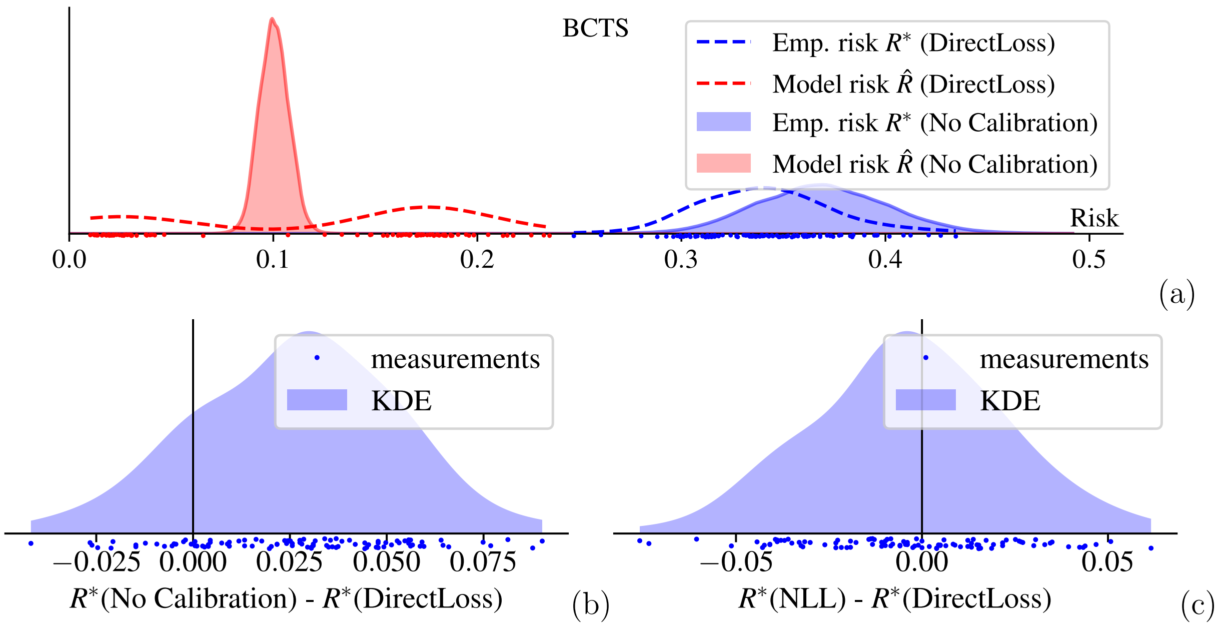

Figure 4.j: The performance of BCTS on Ham10000 dataset, showing the model and true risks before and after calibration. (a) As we increase the number of parameters, it sees that the risk gap increases as compared to TS. (b) The effect of calibration stays positive. (c) Comparing NLL to Direct Loss still gives us no statistically significant results.

perform better by not a large margin, however, it was trained knowing the test time ϵ.

4.4.2 Dangerous class loss varies on CIFAR-100 superclasses

Here, we determine how the calibration methods perform on a problem where we do not know the exact loss for missing the dangerous class, but we know that the loss is between 100 and 1000. The Integral Loss' l_0 matrix will therefore have the loss 100 for missing the class and l_1 will be 900, so that $l_0 + \epsilon l_1$ makes the loss vary between 100 and 1000 based on the ϵ value. The results are in the figure 4.m. We also show the comparison between different calibration methods for the best performing loss criterions in figure 4.n.

In terms of the effect of the loss criterions, from the plots it seems that Integral Loss and NLL outperform the rest. Direct Loss, even though trained with the correct test time ϵ performs substantially worse than the other loss criterions. Nonetheless, we see that calibration improves the performance over no calibration. For TS, Integral Loss is the best performing, but starts getting worse compared to NLL as we increase the number of parameters. The best performing combination of loss criterion and

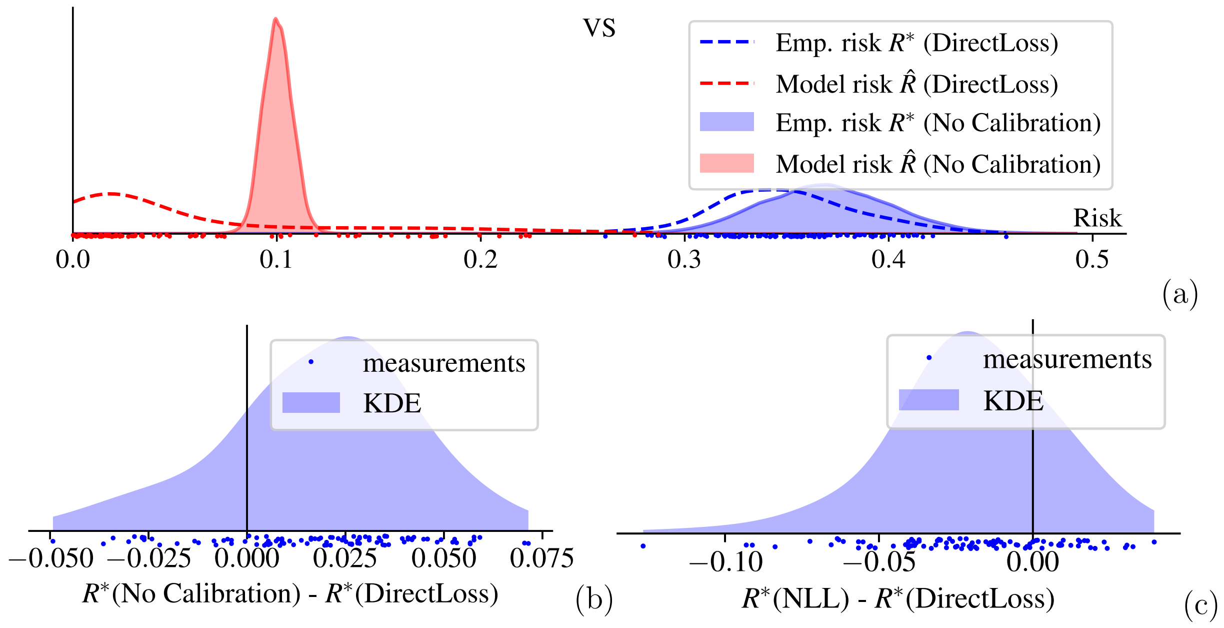

Figure 4.k: The performance of VS on Ham10000 dataset, showing the model and true risks before and after calibration. (a) Having even more parameters than BCTS, the risk gap looks to be worse than without any calibration. (b) We still see a positive effect of calibrating in terms of the actual risk. (c) Here as compared to the other methods, NLL seems to improve the true risk better than Direct Loss.

parametrization is either TS with Integral Loss or VS with NLL, based on the figure 4.n.

4.4.3 Constant unknown loss for the reject option in Ham10000

Lastly, we designed an experiment that should imitate a real-life situation, where a downstream task will show uncertainty in its decision making and ask for professional help. The loss for this will vary, but will be constant with respect to the classes. This "*I do not know*" loss varied between 0 and 9. Results are in figure 4.o.

Noticably, again, as we add parameters, ECE and Integral Loss start performing worse. Direct Loss tends to be outperformed for all parametrizations, while NLL performs well on all of them.

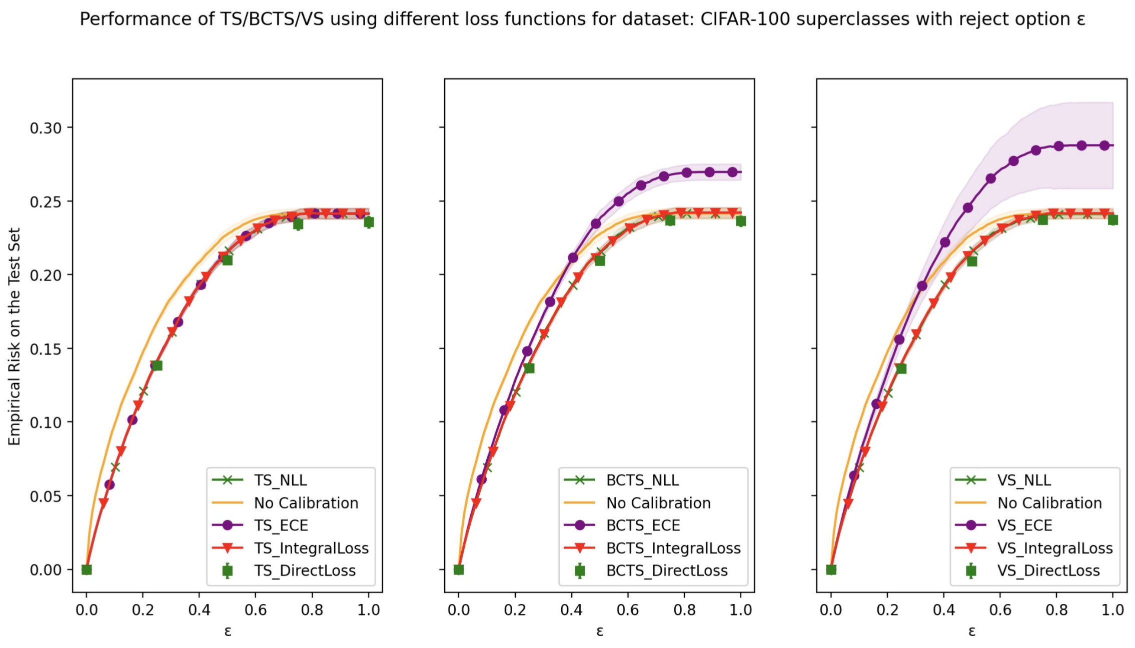

Figure 4.l: Performance of different methods on the confidence calibration problem. NLL and ECE are not aware of the loss. Integral Loss calibrates for $\mathcal{L} = l_0 + \epsilon l_1 \mid \epsilon \in [0, 1]$. Direct Loss calibrates for $l_\epsilon = l_0 + \epsilon l_1$. All methods are tested with Bayes optimal decision strategy (2.18) with l_ϵ. The plots visualize the test performance depending on ϵ.

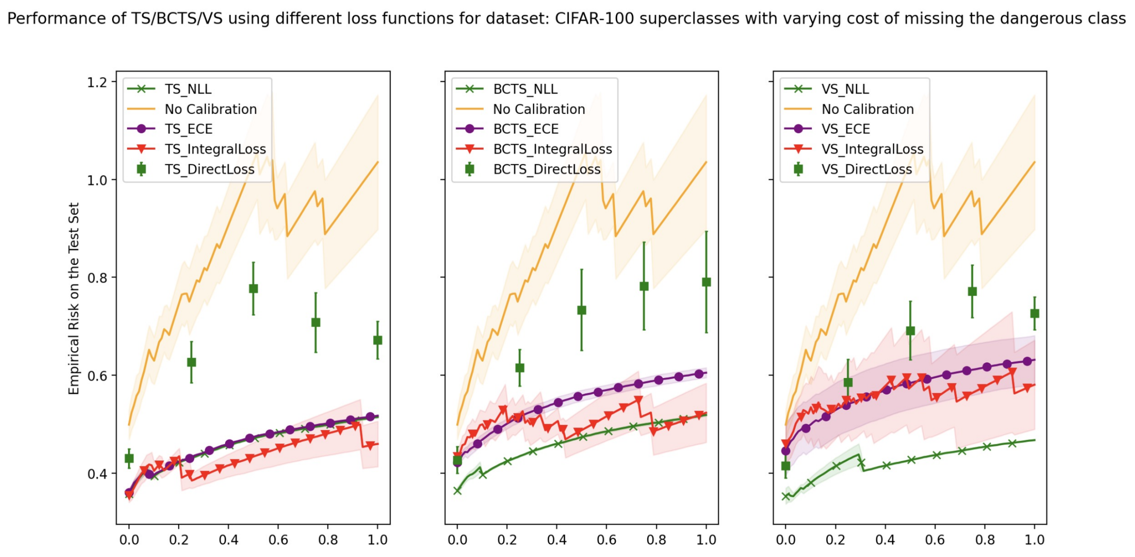

Figure 4.m: Performance of the different parametrizations on the CIFAR–100 superclass version dataset calibration experiment.

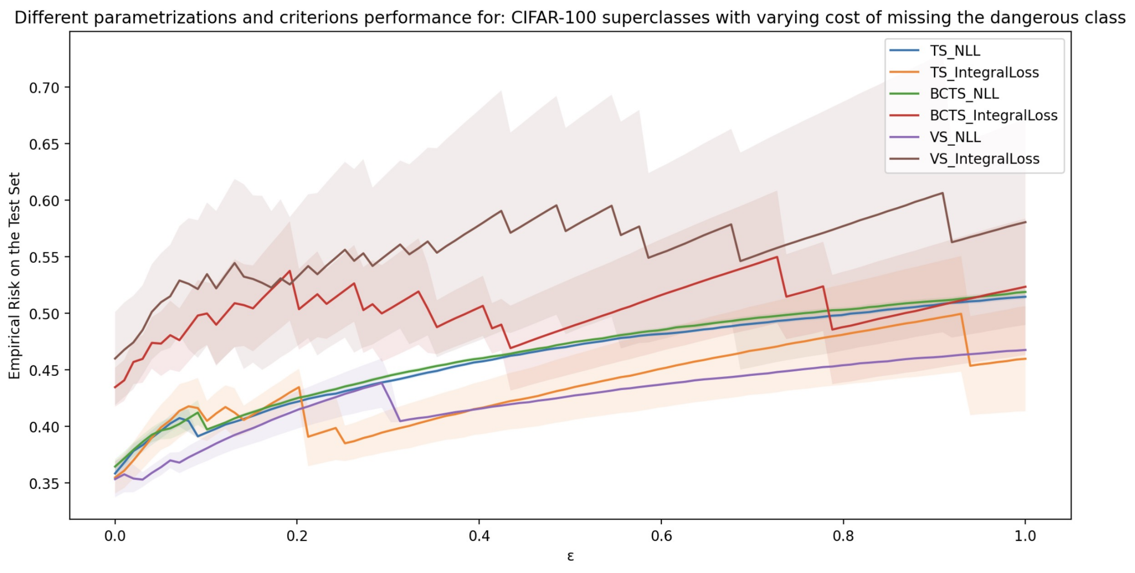

Figure 4.n: The comparison of the different calibration methods with the Integral Loss and the NLL on the CIFAR100 superclass version dataset with varying losses of missing the dangerous class.

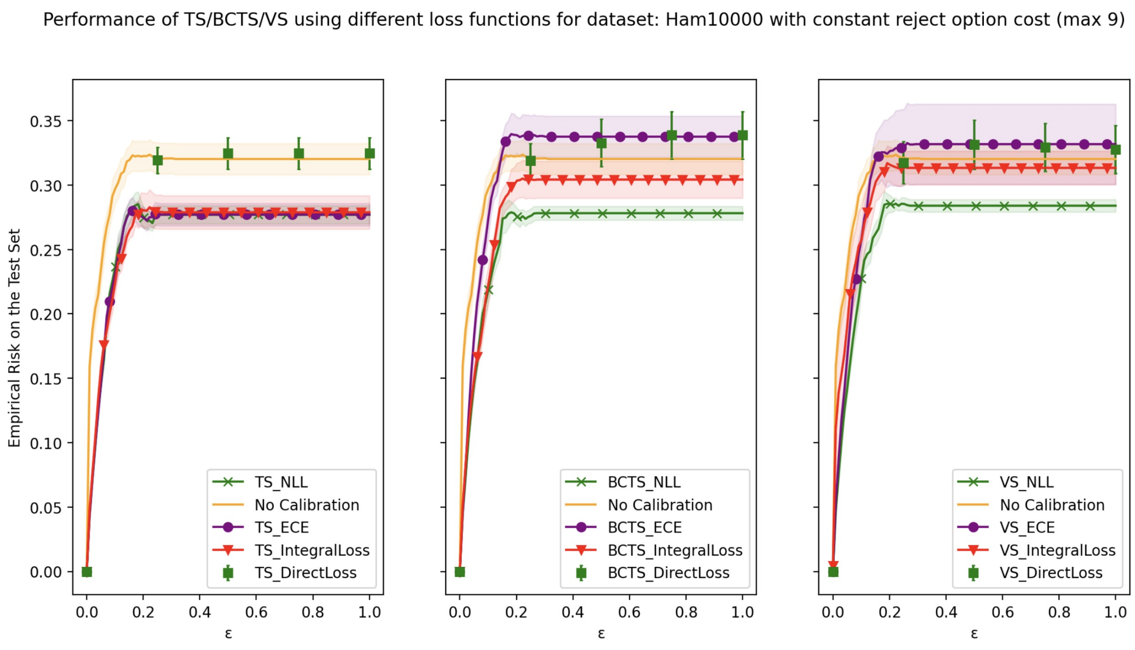

Figure 4.o: The result of different parametrizations on the ham10000 experiment with varied rejection loss.

4.5 Prior shift adaptation for a test prior distribution as a convex combination

As the last experiment, we show the performance of the loss criterions and calibrations for prior shift adaptation. The CIFAR–100 dataset was trained on a uniform distribution of priors. We assume that the test time priors are a convex combination between the uniform distribution and a long tailed distribution, as described in the subsection 3.5.1. As we are assessing the prior shift performance, we assume 0/1 loss. We will use 2 different long-tailed distribution, generated as the normalized power law distribution, i.e.:

$$p(x) = x^{-k},$$

where $k \in \{1, 1.5\}$. We refer to the prior distribution with $k = 1$ as LTk1 – the most common class has a prior probability 28%, and the one with $k = 1.5$ as LTstronger – the most common class is more extreme, having a 46% probability. Further, we sort the classes so that the most common class is the class with the worst accuracy from the trained model for an easier interpretation of the effect of the prior shift adaptation, as the class-wise accuracy is affected by the prior shift of the long-tailed distribution. The experiments are calculated on 10 folds, showing the mean and one standard deviation. We again, plot the results with respect to ϵ – the strength of the long-tailed distribution, where with $\epsilon = 1$ the test prior distribution becomes the long-tailed distribution. Plots are shown in figures 4.p and 4.q.

Clearly, for most of the values of ϵ, any calibration with any parametrization has a positive effect on the performance for both of the new test priors. Direct Loss, again, even with the knowledge of the test time ϵ does not outperform NLL or Integral Loss. On the other hand, our loss criterion Integral Loss does not seem to overfit for the 0/1 loss in any of the parametrizations. Given the standard deviation, we cannot conclude with certainty which loss criterion performs better.

4.6 Discussion

In the experiments, where the loss matrix is known exactly, we show that in the Danish Fungi experiment Direct Loss confidently outperforms other loss criterions. We can also see that using calibration significantly improves the Empirical Risk. For the Ham10000 experiment, we cannot confidently assess whether any of our loss criterions perform better or worse than the baselines (ECE, NLL) and neither can we say whether calibration makes a substantial difference in terms of the Empirical Risk itself. However, we note that calibration can tighten the gap between the true and the model risk. We also notice that as we increase the number of parameters, this gap between the risks widens.

The following experiments, in which we do not know the test time loss matrix exactly, but can assume that it is a convex combination of two matrices that we know, we

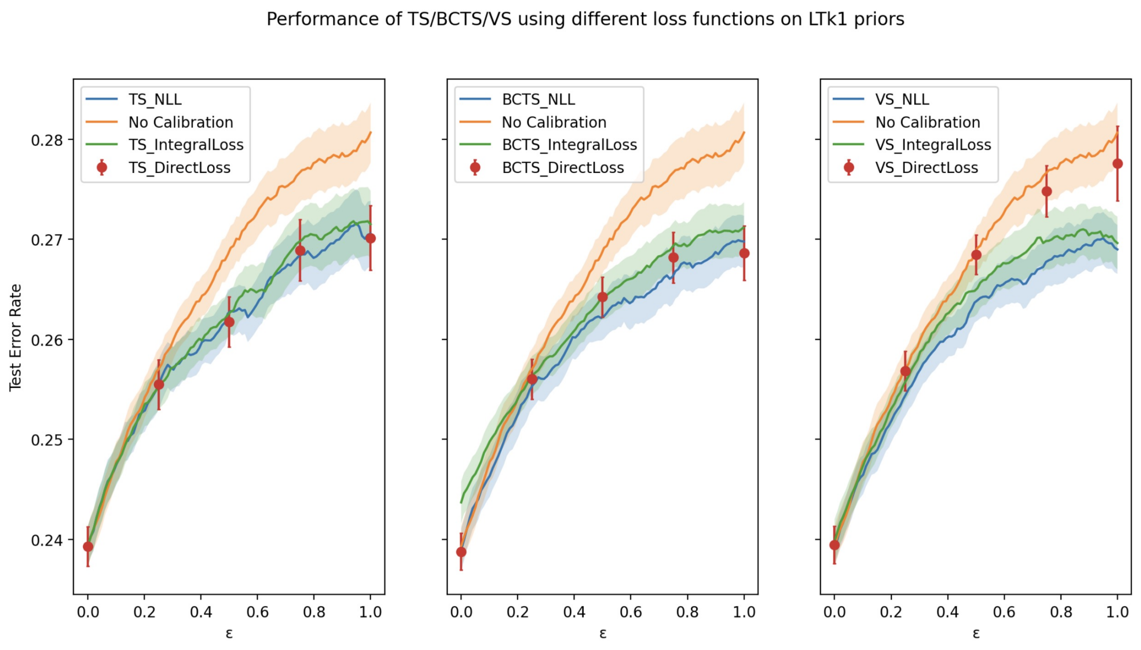

Figure 4.p: The result of the prior shift for different parametrizations on the LTk1 priors.

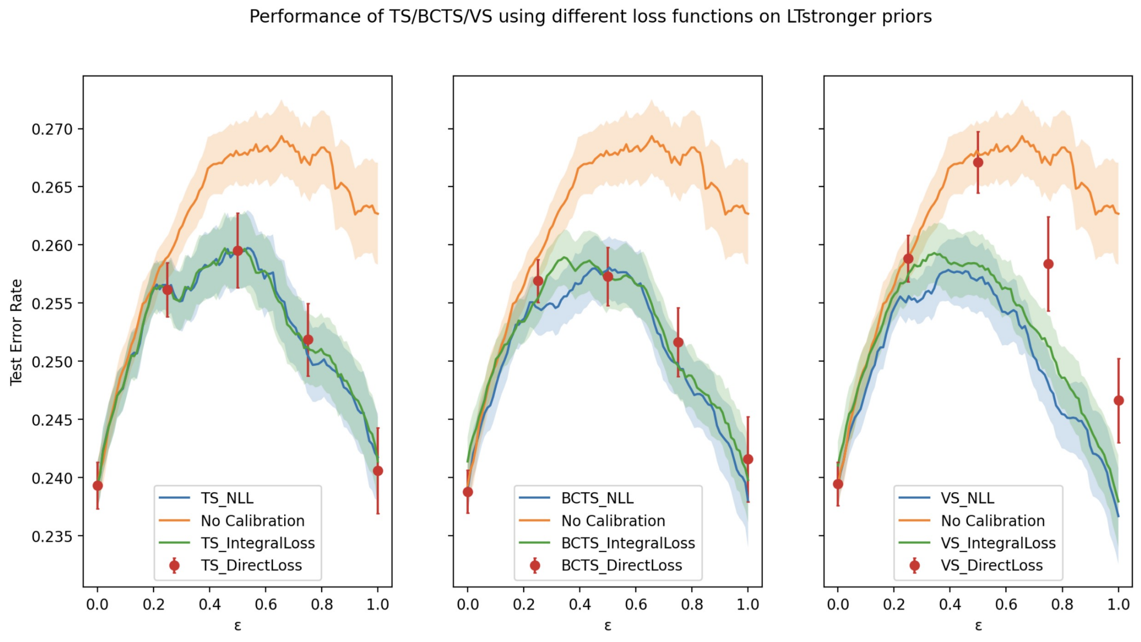

Figure 4.q: The result of the prior shift for different parametrizations on the LTstronger priors.

mainly notice that the as we increase the number of parameters, the performance tends to become worse, however any calibration is usually better than no calibration. In this part, NLL tends to dominate in terms of consistency over Integral Loss – as we increase the number of parameters, the loss does not differ by a lot / does not worsen. In terms of the Empirical Risk we cannot confidently decide, whether

Integral Loss performs better than NLL, given the confidence intervals – we find this interesting as NLL does not take the loss matrix into account, yet their performance is similar. We observe that Integral Loss tends to not generalize that well as the number of parameters increase in these cases. Perhaps, this could be improved using different optimization methods or by using regularizations, etc.

Lastly, in the prior shift adaptation calibration experiments, we find that Integral Loss happens to perform similarly well to NLL, even with the different calibration parametrizations. Direct Loss tends to overfit for BCTS and for VS.

In summary, we were capable of designing loss criterions that are on par with NLL in terms of performance in the tasks that we desgined. We believe, in tasks with more specialized loss matrices, Direct Loss or Integral Loss will outperform NLL, just like in the Danish Fungi experiment. We also show that calibrating the model tends to reduce the Empirical Risk in decision-making.

Chapter 5

Conclusion

The goal of this thesis was to suggest a way to calibrate the outputs of neural networks. We further specialized the task to calibrate the outputs for decision making as multiclass calibration seems to be a very non-trivial task. In addition to that, we were supposed to try and calibrate the outputs so that they perform well under a prior shift.

Throughout this thesis, we firstly give the reader the technical background and align ourselves in the mathematical notation and concepts. Then, we introduce the basic calibration strategies and methods with the motivations behind them. Lastly, we go through some dataset shifts that often appear in applications.

Chapter 3 focuses on explaining our thought processes behind the loss criterions that we proposed. Firstly, we suggest Direct Loss including the forward and backward algorithms and prove that Direct Loss can become an upper or lower bound of the Empirical Risk. Afterwards, we propose Integral Loss, another loss criterion that can be used in similar use cases as Direct Loss. We show the whole derivation of the method and present how Integral Loss can be used for a certain prior shift adaptation situation. Finally, we show the workflow of our experimental process.

In the Experiments chapter, we explain the experiments in more detail and present the results. The experiments include the usage of both of our proposed loss criterions which are compared to NLL and to the situation where no calibration is made. We start by showing some pilot experiments with the loss functions proposed to give the reader some intuition. We proceed with 3 different cases, which are based on what knowledge is one given during calibration time in terms of the test time loss matrix. In each of these experiments we try to present the results with some kind of statistical confidence. We finish by providing a discussion of the results.

5.1 Future Work

We hypothesize that the calibration methods with more parameters if optimized better will improve the performance more significantly using the new criterions. That is because BCTS and VS are generalizations of TS, which has already improved the performance in many of the experiments. As the parameters tend to overfit, perhaps regularization techniques or even different optimization algorithms would help.

For Integral Loss, one obvious extension is to find ways to create a generalization for more than just a convex combination of two matrices. Direct Loss would benefit from decreasing the number of hyperparameter combinations – lowering the need for the hyperparameters or knowing the range of the hyperparameters could improve the performance. Further, although there are essentially no restrictions for the loss matrices, knowing for what loss matrices the different criterions start to deteriorate would extend the project. This could also outline why or when does NLL perform better.

Bibliography

Amr Alexandari, Anshul Kundaje, and Avanti Shrikumar. Maximum likelihood with bias-corrected calibration is hard-to-beat at label shift adaptation. In *International Conference on Machine Learning*, pp. 222–232. PMLR, 2020.

Peter L Bartlett, Michael I Jordan, and Jon D McAuliffe. Convexity, classification, and risk bounds. *Journal of the American Statistical Association*, 101(473):138–156, 2006.

Jochen Bröcker. Reliability, sufficiency, and the decomposition of proper scores. *Quarterly Journal of the Royal Meteorological Society: A journal of the atmospheric sciences, applied meteorology and physical oceanography*, 135(643): 1512–1519, 2009.

Jochen Bröcker and Leonard A Smith. Increasing the reliability of reliability diagrams. *Weather and forecasting*, 22(3):651–661, 2007.

Annabelle Carrell, Neil Mallinar, James Lucas, and Preetum Nakkiran. The calibration generalization gap. *arXiv preprint arXiv:2210.01964*, 2022.

Morris H DeGroot and Stephen E Fienberg. Assessing probability assessors: Calibration and refinement. Technical report, CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT OF STATISTICS, 1981.

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

Sebastian Gregor Gruber and Florian Buettner. Better uncertainty calibration via proper scores for classification and beyond. In *Advances in Neural Information Processing Systems*, 2022.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International Conference on Machine Learning*, pp. 1321–1330. PMLR, 2017.

Pierre Hansen, Brigitte Jaumard, and Gilles Savard. New branch-and-bound rules for linear bilevel programming. *SIAM Journal on scientific and Statistical Computing*, 13(5):1194–1217, 1992.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Tuan Anh Ho, Jiri Matas, and Alexander Shekhovtsov. Calibration for decision making via empirical risk minimization. 2022.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Meelis Kull, Miquel Perello Nieto, Markus Kängsepp, Telmo Silva Filho, Hao Song, and Peter Flach. Beyond temperature scaling: Obtaining well-calibrated multi-class probabilities with dirichlet calibration. *Advances in neural information processing systems*, 32, 2019.

Aviral Kumar, Sunita Sarawagi, and Ujjwal Jain. Trainable calibration measures for neural networks from kernel mean embeddings. In *International Conference on Machine Learning*, pp. 2805–2814. PMLR, 2018.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Matthias Minderer, Josip Djolonga, Rob Romijnders, Frances Hubis, Xiaohua Zhai, Neil Houlsby, Dustin Tran, and Mario Lucic. Revisiting the calibration of modern neural networks. *Advances in Neural Information Processing Systems*, 34:15682–15694, 2021.

Allan H Murphy and Robert L Winkler. A general framework for forecast verification. *Monthly weather review*, 115(7):1330–1338, 1987.

Mahdi Pakdaman Naeini, Gregory Cooper, and Milos Hauskrecht. Obtaining well calibrated probabilities using bayesian binning. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

Alexandru Niculescu-Mizil and Rich Caruana. Predicting good probabilities with supervised learning. In *Proceedings of the 22nd international conference on Machine learning*, pp. 625–632, 2005.

Jeremy Nixon, Michael W Dusenberry, Linchuan Zhang, Ghassen Jerfel, and Dustin Tran. Measuring calibration in deep learning. In *CVPR Workshops*, volume 2, 2019.

Andrew Nobel. Histogram regression estimation using data-dependent partitions. *The Annals of Statistics*, 24(3):1084–1105, 1996.

Anusri Pampari and Stefano Ermon. Unsupervised calibration under covariate shift. *arXiv preprint arXiv:2006.16405*, 2020.

Lukáš Picek, Milan Šulc, Jiří Matas, Thomas S Jeppesen, Jacob Heilmann-Clausen, Thomas Læssøe, and Tobias Frøslev. Danish fungi 2020-not just another image recognition dataset. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 1525–1535, 2022.

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

Roshni Sahoo, Shengjia Zhao, Alyssa Chen, and Stefano Ermon. Reliable decisions with threshold calibration. *Advances in Neural Information Processing Systems*, 34:1831–1844, 2021.

Tomas Sipka, Milan Sulc, and Jiri Matas. The hitchhiker's guide to prior-shift adaptation. *arXiv preprint arXiv:2106.11695*, 2021.

Yang Song, Alexander Schwing, Raquel Urtasun, et al. Training deep neural networks via direct loss minimization. In *International conference on machine learning*, pp. 2169–2177. PMLR, 2016.

Masashi Sugiyama, Matthias Krauledat, and Klaus-Robert Müller. Covariate shift adaptation by importance weighted cross validation. *Journal of Machine Learning Research*, 8(5), 2007.

Christian Tomani, Sebastian Gruber, Muhammed Ebrar Erdem, Daniel Cremers, and Florian Buettner. Post-hoc uncertainty calibration for domain drift scenarios. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10124–10132, 2021.

Philipp Tschandl, Cliff Rosendahl, and Harald Kittler. The ham10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions. *Scientific data*, 5(1):1–9, 2018.

Juozas Vaicenavicius, David Widmann, Carl Andersson, Fredrik Lindsten, Jacob Roll, and Thomas Schön. Evaluating model calibration in classification. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 3459–3467. PMLR, 2019.

Marin Vlastelica, Anselm Paulus, Vít Musil, Georg Martius, and Michal Rolínek. Differentiation of blackbox combinatorial solvers. *arXiv preprint arXiv:1912.02175*, 2019.

David Widmann, Fredrik Lindsten, and Dave Zachariah. Calibration tests in multi-class classification: A unifying framework. *arXiv preprint arXiv:1910.11385*, 2019.

Bianca Zadrozny and Charles Elkan. Transforming classifier scores into accurate multiclass probability estimates. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 694–699, 2002.

Shengjia Zhao, Michael Kim, Roshni Sahoo, Tengyu Ma, and Stefano Ermon. Calibrating predictions to decisions: A novel approach to multi-class calibration. *Advances in Neural Information Processing Systems*, 34, 2021.