UNIVERSITY OF TARTU
Faculty of Science and Technology
Institute of Computer Science
Computer Science Curriculum

Kaspar Valk

Calibration of Multi-Class Probabilistic Classifiers

Master's Thesis (30 ECTS)

Supervisor: Meelis Kull, PhD

Tartu 2022

Calibration of Multi-Class Probabilistic Classifiers

Abstract: Classifiers, machine learning models that predict probability distributions over classes, are not guaranteed to produce realistic output. A classifier is considered calibrated if the produced output is in correspondence with the actual class distribution. Calibration is essential in safety-critical tasks where small deviations between the predicted probabilities and the actual class distribution can incur large costs. A common approach to improve the calibration of a classifier is to use a hold-out data set and a post-hoc calibration method to learn a correcting transformation for the classifier's output. This thesis explores the field of post-hoc calibration methods for classification tasks with multiple output classes: several existing methods are visualized and compared, and three new non-parametric post-hoc calibration methods are proposed. The proposed methods are shown to work well with data sets with fewer classes, managing to improve the state-of-the-art in some cases. The basis of the three suggested algorithms is the assumption of similar calibration errors in close neighborhoods on the probability simplex, which has been previously used but never clearly stated in the calibration literature. Overall, the thesis offers additional insight into the field of multi-class calibration and allows for the construction of more trustworthy classifiers.

Keywords: machine learning, multi-class, classifier, calibration

CERCS: P176 Artificial intelligence

Mitmeklassiliste tõenäosuslike klassifitseerijate kalibreerimine

Lühikokkuvõte: Klassifitseerijad ehk masinõppe mudelid, mis ennustavad klasside tõenäosusjaotust, ei ole tagatud väljastama realistlikke tõenäosuseid. Klassifitseerijat peetakse kalibreerituks, kui selle väljund on vastavuses tegeliku klassijaotusega. Kalibreeritus on eriti oluline ohutust nõudvate ülesannete puhul, kus väikesed kõrvalekalded ennustatud tõenäosuste ja tegeliku klassijaotuse vahel võivad põhjustada suurt kahju. Tavapärane lähenemine klassifitseerijate kalibreerituse parandamiseks on kasutada valideerimisandmestikku ja kalibreerimismeetodit, et õppida klassifitseerija väljundit korrigeeriv teisendus. See lõputöö uurib mitme väljundklassiga klassifitseerijate kalibreerimismeetodeid: mitmeid olemasolevaid meetodeid visualiseeritakse ja võrreldakse; ning pakutakse välja kolm uut mitteparameetrilist kalibreerimismeetodit. Töös näidatakse, et väljapakutud meetodid töötavad hästi vähemate klassidega andmestikel, suutes mõnel korral ületada tugevamaid konkurente. Kõigi kolme väljapakutud algoritmi aluseks on eeldus, et kalibreerimisviga on tõenäosusruumi lähedastes piirkondades sarnane — eeldus, mida on varasemas kalibreerimiskirjanduses kasutatud, kuid mida pole kunagi selgelt sõnastatud. Kokkuvõtvalt pakub lõputöö täiendavat ülevaadet mitmeklassiliste

mudelite kalibreerimisest ning võimaldab koostada usaldusväärsemaid klassifitseerijaid.

Võtmesõnad: masinõpe, mitmeklassiline, klassifitseerija, kalibreerimine

CERCS: P176 Tehisintellekt

Contents

1 Introduction

A classifier is a machine learning model that predicts the class of an input data point. For example, a classifier could be tasked to predict whether the weather tomorrow at noon in Tartu will be *sunny* or *not sunny* given the current weather observation data; or whether an X-ray image of a person's chest depicts a *healthy lung*, *lung cancer* or *some other lung disease*. In many classification tasks, it is more valuable to use a probabilistic classifier that is able to predict a probability distribution over classes, not just one class value. For example, in weather forecasting, where uncertainties are inherent, the probability of rain is more desirable than a binary yes-no prediction.

It can be possible, however, that a probabilistic classifier is not well-calibrated and produces distorted probabilities. A classifier is considered to be calibrated if its predicted probabilities are in correspondence with the true class distribution. For example, predicting an X-ray image to show a *healthy lung* with a probability of 0.9 is calibrated if, among a large sample of images with similar predictions, 0.9 of them truly depict a healthy lung. If in reality only 0.7 of the images depict a healthy lung, then the prediction of 0.9 is over-confident. The problem of over-confident predictions is especially common for modern deep neural networks [8, 22]. Distorted output probabilities are also characteristic of many classical machine learning methods such as naive Bayes or decision trees [33, 7].

Well-calibrated classifiers are essential in safety-critical applications (e.g. medical diagnosis, autonomous driving) where small differences between predicted probabilities and the true label distribution can cause costly mistakes [23]. For example, in a self-driving car that uses a classifier to detect if the road is clear of obstructions, over-confident predictions can lead to accidents, and under-confident predictions can prevent the vehicle from driving.

The machine learning literature has two fundamentally different approaches to achieve better-calibrated classifiers. The first approach, with a focus on neural networks, is to modify the classifier's training algorithm or use Bayesian approaches to model uncertainties. For example, Mukhoti et al. [27] studied the use of focal loss [24] instead of log-loss for training better calibrated classifiers; Müller et al. [28] investigated the use of label smoothing [40] during training for better calibration; Kumar et al. [21] and Popordanoska et al. [37] proposed additional terms to be added to the training time loss function that penalize miscalibration; Maddox et al. [25] proposed Bayesian model averaging for achieving calibration in deep learning.

The second approach to achieve better-calibrated classifiers is to apply a *post-hoc calibration method* on an already trained classifier. Post-hoc calibration methods receive as input a classifier and a hold-out calibration data set and learn a transformation from the classifier's predictions to better-calibrated predictions. Many methods have been proposed for binary probabilistic classifiers, where the output has only two classes and only one degree of freedom. For example, there exists logistic calibration [36]; isotonic

calibration [43]; histogram binning [42]; beta calibration [18]. For multi-class classifiers with more than two output classes, a common approach has been to apply binary methods in a one-versus-rest manner: a binary post-hoc calibration method is applied separately on the probabilities of each class [43]. In recent years several inherently multi-class post-hoc calibration methods have been proposed as well, even though some of them are applicable only for neural networks. For example, Guo et al. [8] proposed temperature scaling, vector scaling, and matrix scaling; Kull et al. [19] introduced Dirichlet calibration; Zhao et al. [46] proposed a method specifically intended for decision making scenarios; Rahimi et al. [38] suggested intra-order preserving functions.

This thesis takes a look at the *post-hoc calibration method* approach for achieving better calibrated multi-class classifiers and proposes several intuitive non-parametric methods for the task. The basis of the proposed methods is assuming close predictions on the probability simplex to have similar calibration errors.

In Section 2, notation is introduced and an overview of background information connected to multi-class calibration is given. The concepts of calibration, calibration error, calibration error estimation, and existing post-hoc calibration methods for multi-class calibration are explained. In Section 3, three new non-parametric post-hoc calibration methods are proposed. In Section 4, experiments are carried out to compare the proposed methods to their competitors.

2 Background

The following sections introduce the concepts of calibration, calibration error, calibration error estimation and explain the existing post-hoc calibration methods for multi-class calibration.

2.1 Notation

This thesis focuses on a m-class supervised classification problem with feature space $\mathcal{X}$ and label space $\mathcal{Y} = \{(1, 0, \ldots, 0), (0, 1, \ldots, 0), \ldots, (0, 0, \ldots, 1)\}$, where $m \in \mathbb{N} \setminus \{0, 1, 2\}$ and the labels are one-hot encoded. A probabilistic multi-class classifier for such a classification problem is a function $\boldsymbol{f} : \mathcal{X} \to \Delta^m$ that takes as input features $\boldsymbol{x} \in \mathcal{X}$ and outputs a probability vector $\boldsymbol{f}(\boldsymbol{x}) = \hat{\boldsymbol{p}} = (\hat{p}_1, \ldots, \hat{p}_m) \in \Delta^m$, where $\Delta^m = \{(q_1, \ldots, q_m) \in [0, 1]^m | \sum_{i=1}^{m} q_i = 1\}$ is the $(m-1)$-dimensional probability simplex. In addition, let $\boldsymbol{X} \in \mathcal{X}$, $\boldsymbol{Y} = (Y_1, \ldots, Y_m) \in \mathcal{Y}$ and $\boldsymbol{f}(\boldsymbol{X}) = \hat{\boldsymbol{P}} = (\hat{P}_1, \ldots, \hat{P}_m) \in \Delta^m$ be random variables, where $\boldsymbol{X}$ denotes the input features, $\boldsymbol{Y}$ denotes the label, and $\hat{\boldsymbol{P}}$ the classifier's prediction.

In Figure 1 is depicted a 2-dimensional probability simplex for a 3-class classification problem. The simplex contains all the possible probability vector values for 3 classes. Each axis depicts the probability of one of the classes.

2.2 Calibration

A classifier is considered to be calibrated if the probabilities it outputs are in correspondence with reality. The notion of being calibrated requires that the probability of observing the predicted classes is equal to the predicted probability of the classifier. There exist several exact definitions for calibration in the context of probabilistic multi-class classifiers. In this thesis, the most common definitions of multi-class, classwise, and confidence calibration are used, but some less common definitions of top-label, top-k-confidence, and top-k-label calibration are also introduced for background.

Multi-class calibration. The most strict notion of calibration requires every possible output vector to match reality [5]. A classifier is considered to be multi-class-calibrated (or just calibrated) if for any prediction vector $\hat{\boldsymbol{p}} \in \Delta^m$ it holds that

$$\mathbb{E}\left[\boldsymbol{Y} | \hat{\boldsymbol{P}} = \hat{\boldsymbol{p}}\right] = \hat{\boldsymbol{p}}.$$

According to multi-class calibration, among all data points for which the classifier has predicted for example $(0.6, 0.3, 0.1)$, proportionally 0.6 of them have to belong to class 1, 0.3 to class 2, and 0.1 to class 3.

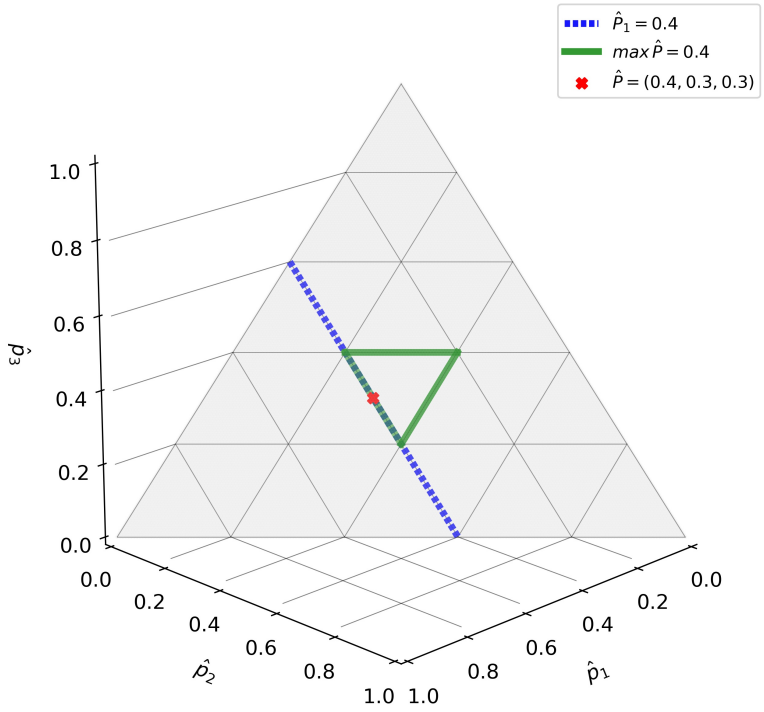

Figure 1. An example of different ways to condition the expectation for the multi-class, classwise, and confidence calibration definitions on a 2-dimensional probability simplex.

Classwise calibration. A weaker notion of classwise calibration requires the predictions to match reality when looking at each class separately [43]. A classifier is considered to be classwise calibrated if for any class $i \in \{1, 2, \ldots, m\}$ and any real value $c \in [0, 1]$ it holds that

$$\mathbb{E}\left[Y_i | \hat{P}_i = c\right] = c.$$

Note that for binary classification, classwise calibration is the same as multi-class calibration [41].

Confidence calibration. Confidence calibration is another weaker notion introduced by Guo et al. [8], that requires only the predictions for the class with the highest probability in each output to match reality. A classifier is considered to be confidence calibrated if for any real value $c \in [0, 1]$ it holds that

$$\mathbb{E}\left[Y_{argmax\,\hat{\boldsymbol{P}}} | max\,\hat{\boldsymbol{P}} = c\right] = c.$$

Top-label calibration. A recent notion of top-label calibration proposed by Gupta and Ramdas [9] that can be viewed as confidence calibration applied classwise, requires the predictions for the class with the highest probability in each output to match reality but conditioned on each class separately. A classifier is considered to be top-label calibrated if for any class $i \in \{1, 2, \ldots, m\}$ and any real value $c \in [0, 1]$ it holds that

$$\mathbb{E}\left[Y_{argmax\,\hat{\boldsymbol{P}}} | max\,\hat{\boldsymbol{P}} = c \wedge argmax\,\hat{\boldsymbol{P}} = i\right] = c.$$

Top-k-confidence calibration. Top-k-confidence calibration proposed by Gupta et al. [10] is a stricter version of confidence calibration, requiring the predictions for the class with the highest, second-highest, and up to the k-th highest probability in each output to match reality. A classifier is considered to be top-k-confidence calibrated if for any $j \in \{1, 2, \ldots k\}$ and any real value $c \in [0, 1]$ it holds that

$$\mathbb{E}\left[Y_{\underset{j}{argmax}\,\hat{\boldsymbol{P}}} | \underset{j}{max}\,\hat{\boldsymbol{P}} = c\right] = c,$$

where $\underset{j}{max}\,\hat{\boldsymbol{P}}$ denotes the j-th largest value of the output and $\underset{j}{argmax}\,\hat{\boldsymbol{P}}$ denotes the class of the j-th largest value of the output.

Top-k-label calibration. Top-k-label calibration proposed by Gupta and Ramdas [9] is a stricter version of top-label calibration, requiring the same conditions as top-k-confidence calibration but in addition conditioned on each class separately. A classifier is considered to be top-k-label calibrated if for any class $i \in \{1, 2, \ldots m\}$, any $j \in \{1, 2, \ldots k\}$ and any real value $c \in [0, 1]$ it holds that

$$\mathbb{E}\left[Y_{\underset{j}{argmax}\,\hat{\boldsymbol{P}}} | \underset{j}{max}\,\hat{\boldsymbol{P}} = c \wedge \underset{j}{argmax}\,\hat{\boldsymbol{P}} = i\right] = c.$$

Note that there is some confusion in the literature about naming the different definitions of calibration. Following Kull et al. [19], the names multi-class and classwise calibration have been used. However, multi-class calibration has previously also been known as *reliability* [5] or as *validity* [6]; classwise calibration has existed for some time without a proper name since first introduced by Zadrozny and Elkan [43], but has also been called *marginal calibration* [20]. Confidence calibration has been called *top-label calibration* in some works [20, 45].

The definitions of calibration mainly differ by what the expectations are conditioned on: how are the predicted probabilities aggregated among which the prediction value must match reality. In Figure 1 three specific examples are shown on what the expectation could be conditioned on: one for each of the three common definitions of calibration of multi-class, classwise, and confidence, to help visualize the differences between

the definitions. The figure depicts different aggregations for a 3 class problem when conditioned on the prediction vector $\hat{\boldsymbol{P}} = (0.4, 0.3, 0.3)$; the prediction value $\hat{P}_1 = 0.4$ for class 1; and the maximum predicted value $max\ \hat{\boldsymbol{P}} = 0.4$. Note that new definitions of calibration could easily be defined by redefining on what to condition the expectation on and on which classes of the label vector to take the expectation.

2.3 Calibration error

Calibration error describes the difference between the predicted probabilities of the classifier and the true class probabilities. There are several scenarios in which it is desirable to know a classifier's calibration error:

- in a safety-critical application, it can be important to know the general level of trust of the model or some prediction;
- when choosing between several classifiers for a given task, calibration error allows to rank the models based on calibration;
- when developing novel post-hoc calibration methods, it is needed to be able to rank the competing methods.

In this work calibration error is used for the third scenario: to be able to tell if the proposed methods perform any better than their competitors.

Kumar et al. [20] defined calibration error for confidence and classwise calibration for a classifier $\boldsymbol{f}$. In a slightly more generalized form, confidence calibration error is defined as

$$CE_{conf} = \mathbb{E}\left[\left|max\ \hat{\boldsymbol{P}} - \mathbb{E}\left[Y_{argmax\ \hat{\boldsymbol{P}}} | max\ \hat{\boldsymbol{P}}\right]\right|^{\alpha}\right]^{1/\alpha},$$

and classwise calibration error is defined as

$$CE_{cw} = \frac{1}{m}\sum_{i=1}^{m} \mathbb{E}\left[\left|\hat{P}_i - \mathbb{E}\left[Y_i | \hat{P}_i\right]\right|^{\alpha}\right]^{1/\alpha}.$$

The calibration errors are parameterized by α, where $\alpha = 1$ results in mean-absolute-error, and $\alpha = 2$ mean-squared-error.

Calibration error could not only be defined for the whole classifier $\boldsymbol{f}$ but also for just one prediction value as the difference between the right-hand side and the left-hand side of the corresponding calibration definition. For this work, calibration error for multi-class calibration for prediction vector value $\hat{\boldsymbol{p}}$ is defined as

$$CE(\hat{\boldsymbol{p}}) = \hat{\boldsymbol{p}} - \mathbb{E}\left[\boldsymbol{Y} | \hat{\boldsymbol{P}} = \hat{\boldsymbol{p}}\right].$$

Note that for multi-class calibration, error defined in such a way is a vector of values.

2.4 Calibration evaluation

True calibration error can not be directly found in practice. For a data point $(\boldsymbol{x}, \boldsymbol{y})$, the classifier's prediction $\boldsymbol{f}(\boldsymbol{x}) = \hat{\boldsymbol{p}}$ is known, but the expected label value $\mathbb{E}[\boldsymbol{Y}|\hat{\boldsymbol{P}} = \hat{\boldsymbol{p}}]$ is not. Every data point gives only a single realization of the random variable, not the expectation. Therefore, calibration error according to its definition can only be estimated in any real-world setting.

2.4.1 Reliability diagrams and expected calibration error

For confidence calibration and classwise calibration, *reliability diagrams* [29, 33] and *expected calibration error*[1] (ECE) [31] are used to estimate the calibration error on the test set. For multi-class calibration error, there does not exist a commonly used way to estimate it. Reliability diagrams and ECE group similar predictions together and use the average label value of the group as estimates of the expected label value $\mathbb{E}\left[Y_i|\hat{P}_i\right]$ or $\mathbb{E}\left[Y_{argmax\,\hat{\boldsymbol{P}}}|max\,\hat{\boldsymbol{P}}\right]$.

Reliability diagrams. To get visual estimates of calibration, reliability diagrams are used [29, 33]. First, the predictions and labels of a single class are grouped into bins $B_1, \ldots, B_b$ where each bin contains a certain region of probabilities. Then in each bin B_i, the average prediction $\overline{p}_i = \frac{1}{|B_i|}\sum_{\hat{p}_j \in B_i} \hat{p}_j$ and average label value $\overline{y}_i = \frac{1}{|B_i|}\sum_{y_j \in B_i} y_j$ is calculated, where $|B_i|$ denotes the number of predictions in the bin. Finally, the averages are used to draw the diagram; an example is depicted in Figure 2. The example is constructed on synthetically generated predictions for one class; exact details of the data generation are not important. The average label value in each bin determines the height of the bin. The difference between the two averages $\overline{y}_i - \overline{p}_i$ is used as a calibration error estimate in each bin: this is depicted in Figure 2 as the red bars. The average prediction value in each bin determines the x-axis location of the red bar: usually, it happens to be around the bin center due to averaging. With perfect calibration, the two averages $\overline{y}_i$ and $\overline{p}_i$ should approximately match: the tops of the bins should touch the black dotted line and there should be no visible red bars. The reliability diagram in Figure 2 hints at an overconfident classifier: the bars are below the diagonal at high predicted probabilities. This means that the predictions are on average higher than the true probabilities of observing the corresponding label. For example, it can be seen from the reliability diagram that around the predicted probability of 0.85, the actual class proportion is roughly 0.7.

[1] Commonly known as *expected calibration error*, but as proposed by Roelofs et al. [39] it is better named *estimated calibration error* to avoid confusion with the definition of calibration error which includes an expectation

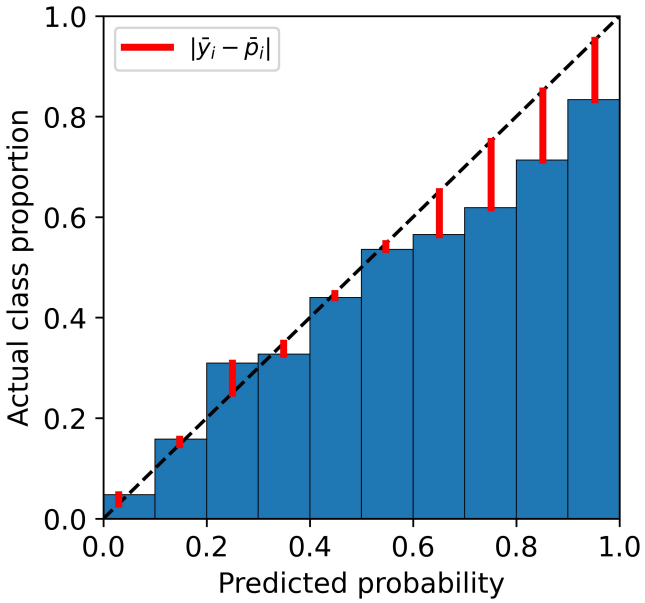

Figure 2. Reliability diagram example.

Expected calibration error. To condense the calibration information depicted in a reliability diagram into one number, the average length of the red bars is used. This measure is called *expected calibration error* (ECE) [31] and it is defined as

$$ECE = \frac{1}{b}\sum_{i=1}^{b}\frac{|B_i|}{n}\cdot|\overline{y}_i - \overline{p}_i|^{\alpha},$$

where n is the total number of data points in the test set and b the number of bins in the binning scheme. As the number of data points in each bin might not be equal, a weighting term $\frac{|B_i|}{n}$ is also used in ECE calculation. The ECE value could measure the average absolute length of the red bars if $\alpha = 1$; or the average squared length of the red bars if $\alpha = 2$. For confidence calibration, the largest predictions $max\,\hat{\boldsymbol{P}}$ and their corresponding labels $Y_{argmax\,\hat{\boldsymbol{P}}}$ are used to draw the reliability diagram and calculate the corresponding ECE value [31]. For classwise calibration, the reliability diagram is constructed, and ECE is calculated separately for each class i on the predictions of the i-th class $\hat{P}_i$ and their corresponding labels Y_i. The final classwise ECE value is considered the arithmetic average ECE value across all the classes [19].

Binning scheme. The binning scheme of reliability diagrams and ECE can be chosen freely but commonly around 15 bins with equal width or equal size are used [41]. Equal width means each bin occupies an equal chunk of the probability space; equal size means

the bins all contain an equal number of data points. In Figure 2, 10 equal width bins are used.

2.4.2 Proper scoring rules

While ECE is an important measure for calibration evaluation, it should not be the only metric to be evaluated. Very low calibration error can be achieved if the classifier makes very uninformative predictions; e.g. predicts the overall class distribution of the training data set for any given input [19]. Proper scoring rules are good measures to consider in addition to ECE [19].

Scoring rules are used to measure the goodness of probabilistic predictions; they are considered proper if and only if they are minimized by predicting the true underlying label distribution [30]. Even though the minimum is obtained by a strictly multi-class calibrated classifier, proper scoring rules alone are not sufficient to evaluate calibration. This is due to the fact that they do not evaluate only calibration error, but also additional terms [17]. For example, proper scoring rules can be shown to decompose into calibration loss (loss due to miscalibration), grouping loss (loss due to several instances given the same prediction $\hat{\boldsymbol{p}}$ but having different true label distributions) and irreducible loss (loss due to inherent uncertainty in the classification task) [17]. Therefore, two scores $s_1 < s_2$ from a proper scoring rule on two different sets of predictions do not imply that the calibration error for the first set of predictions is smaller: the difference might come from grouping loss. In addition, different proper scoring rules can rank two models differently: one scoring rule might produce $s_1 < s_2$ while a different scoring rule might produce $s_1 > s_2$ [26]. The latter problem of differing rankings is not only characteristic of proper scoring rules: it has been shown that the same problem exists for the ECE measure which is sensitive to the choice of binning [2].

Two commonly used proper scoring rules are Brier score [4]

$$BS = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{m}(\hat{p}_{ij} - y_{ij})$$

and log-loss [30]

$$LL = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{m} y_{ij}\log\hat{p}_{ij},$$

where m is the number of classes and n the number of data points in a test set with classifier predictions $\hat{\boldsymbol{p}}_1 = (\hat{p}_{11}, \ldots, \hat{p}_{1m}), \ldots, \hat{\boldsymbol{p}}_n = (\hat{p}_{n1}, \ldots, \hat{p}_{nm})$ and one-hot encoded labels $\boldsymbol{y}_1 = (y_{11}, \ldots, y_{1m}), \ldots, \boldsymbol{y}_n = (y_{n1}, \ldots, y_{nm})$.

2.5 Post-hoc calibration methods

Calibration of an already trained classifier can be improved by a post-hoc calibration method. Given a classifier and a hold-out validation data set different from the original training data set, the goal of a post-hoc calibration method is to learn a map $\hat{\boldsymbol{c}} : \Delta^m \to \Delta^m$ from the uncalibrated output $\hat{\boldsymbol{p}}$ of the classifier to better-calibrated output $\hat{\boldsymbol{c}}(\hat{\boldsymbol{p}})$. The ideal result would be if the calibration method learns the true calibration map $\boldsymbol{c}(\hat{\boldsymbol{p}}) = \mathbb{E}\left[\boldsymbol{Y}|\hat{\boldsymbol{P}} = \hat{\boldsymbol{p}}\right]$ which sets every prediction value equal to the corresponding true label distribution [19]. The transformation is learned by optimizing a proper scoring rule (Brier score or log-loss) [19]. The process of using a post-hoc calibration method is illustrated in Figure 3. In this thesis, the hold-out validation data set is also called the calibration data set.

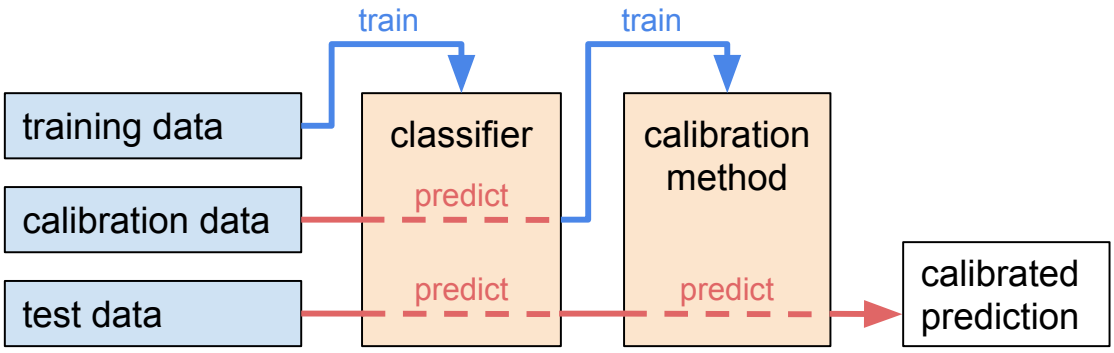

Figure 3. Schematic visualisation of using a post-hoc calibration method.

Logits versus probabilities. Some post-hoc calibration methods are designed specifically for neural networks and are applied on the logits, not the actual probabilities the classifier outputs [8]. Logits $\boldsymbol{z} = (z_1, \ldots, z_m)$ are the values of the final hidden layer in a neural network that are transformed into class probabilities $\hat{\boldsymbol{p}}$ with the softmax function $\boldsymbol{\sigma} : \mathbb{R}^m \to \Delta^m$ [3]. The softmax function $\boldsymbol{\sigma}(\boldsymbol{z}) = \hat{\boldsymbol{p}} = (\sigma_1(\boldsymbol{z}), \ldots, \sigma_m(\boldsymbol{z}))$ is defined to be

$$\sigma_i(\boldsymbol{z}) = \frac{\exp(z_i)}{\sum_{j=1}^{m} \exp(z_j)},$$

for $i = 1, \ldots, m$. It takes as input a vector of real values and transforms them into a probability distribution with probabilities proportional to the exponential values of the input. For methods applied on logits, the calibration function $\hat{\boldsymbol{c}} : \mathbb{R}^m \to \Delta^m$ instead takes as input the logits $\boldsymbol{z}$ not the predictions $\hat{\boldsymbol{p}}$.

Binary versus multi-class. In addition to differentiating between post-hoc calibration methods applied on logits or on probabilities, it is also important to distinguish between

methods for binary and multi-class classifiers. In a binary classification task, there is only one degree of freedom: it is enough to predict only the value of the positive class $\hat{p}$ as the probability of the negative class is defined by $1 - \hat{p}$. Therefore, for binary post-hoc calibration methods, the task is one-dimensional and the calibration method is a function $\hat{c} : \mathbb{R} \to \mathbb{R}$. In the multi-class case, the calibration function has multi-dimensional input and output. Because of this binary post-hoc calibration methods are not directly usable for multi-class classifiers while multi-class methods can be used for binary problems. In the following subsections, the existing methods for multi-class calibration are described.

2.5.1 One-versus-rest methods

A common approach to multi-class calibration has been to apply binary post-hoc calibration methods in a one-versus-rest manner [43]. For every class in a m class classification task, one can define a binary one-vs-rest classification problem: the currently viewed class is the positive class, rest of the classes grouped together are the negative class. In a one-versus-rest approach to multi-class calibration, a binary classification method is trained on m such one-vs-rest tasks separately. There are two main flaws to the one-versus-rest approach: first, it is not able to learn any dependencies between classes; second, when the output of m binary methods is put back together, the prediction vector will likely no longer sum to 1 and needs to be normalized. Three binary calibration methods that have been commonly applied in the one-versus-rest approach are isotonic calibration [43], histogram binning [42] and beta calibration [18].

Isotonic calibration. Zadrozny and Elkan [43] proposed to use isotonic regression for calibration. It learns a piece-wise constant monotone function by minimizing the squared error between the predictions $\hat{p}$ and their labels y.

Histogram binning. Histogram binning is another non-parametric method proposed by Zadrozny and Elkan [42]. It uses a similar strategy to the ECE algorithm described in Section 2.4.1: it divides the probability space into bins and in each bin sets the calibrated value of predictions belonging to that bin equal to the average label value in the bin. Similar to the ECE algorithm the binning scheme can be chosen freely.

Beta calibration. Kull et al. [18] proposed a parametric approach that assumes the classifier predictions can be described with the beta distribution. The post-hoc calibration method is defined by the parametric family

$$\hat{c}(\hat{p}) = \frac{1}{1 + 1 \,/\, \left(e^{d} \frac{\hat{p}^{a}}{(1-\hat{p})^{b}} \right)},$$

where a, b, d are the function parameters learned on the calibration data set.

An illustrative example of the three binary post-hoc calibration methods applied on a synthetic binary calibration task is depicted in Figure 4. The goal of the binary calibration task is to estimate the true calibration function from the classifier predictions and labels in the calibration data set. The exact details of the synthetic data for this figure are not important: it serves to give basic visual intuition of the shapes of the calibration maps produced by the three binary methods.

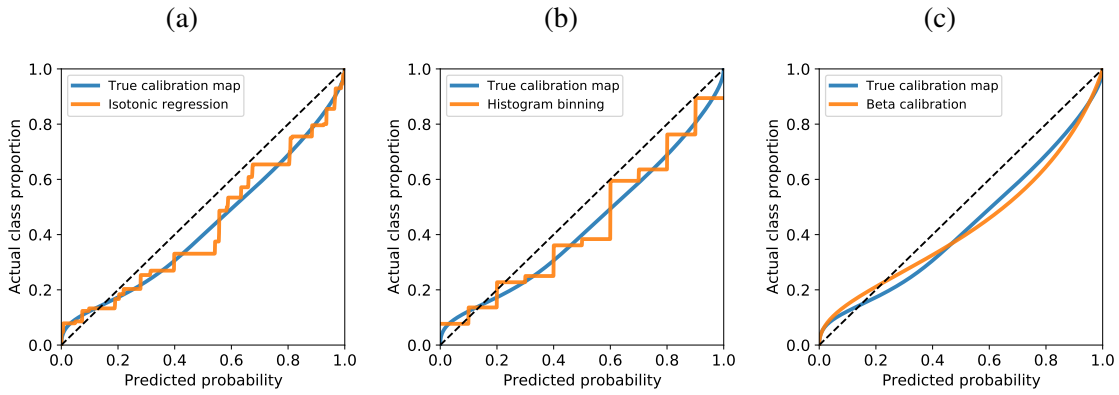

Figure 4. Illustrative example of binary post-hoc calibration methods applied on a synthetic binary calibration task with 3000 data points: (a) isotonic regression; (b) histogram binning with 10 equal-width bins; (c) beta calibration.

2.5.2 Temperature scaling

Temperature scaling is a logit scaling method designed for neural networks introduced by Guo et al. [8]. The method is defined as $\hat{\boldsymbol{c}}(\boldsymbol{z}) = \boldsymbol{\sigma}(\boldsymbol{z}/t)$ where $t \in (0, \infty)$ is the learned temperature parameter shared across all classes. If $t > 1$, then the method makes the predictions less confident by pulling the probability distribution towards the uniform distribution; if $t < 1$, then the method makes the predictions more confident, making the largest probability in the output even larger. For example, consider a 3-class logit vector $\boldsymbol{z} = (6, 4, 2)$ and a temperature parameter $t = 2$. The softmax of the original logit vector is $\boldsymbol{\sigma}(\boldsymbol{z}) \approx (0.87, 0.12, 0.02)$; the softmax of the scaled logit vector is $\boldsymbol{\sigma}(\boldsymbol{z}/2) = \boldsymbol{\sigma}((3, 2, 1)) \approx (0.67, 0.24, 0.09)$. The output becomes less confident: the largest value in the prediction vector is reduced from 0.87 to 0.67 by increasing the predicted values for other classes. With temperature $t = 0.5$, the scaled logits would be $\boldsymbol{\sigma}(\boldsymbol{z}/0.5) = \boldsymbol{\sigma}((12, 8, 4)) \approx (0.98, 0.02, 0.00)$. The output becomes more confident: the largest value in the prediction vector is increased from 0.87 to 0.98 by further decreasing the predicted values for other classes.

2.5.3 Matrix and vector scaling

Matrix and vector scaling are both logit transformation techniques proposed by Guo et al. [8] similar to temperature scaling. The calibrated output of these techniques is obtained by $\hat{\boldsymbol{c}}(\boldsymbol{z}) = \boldsymbol{\sigma}(\mathbf{W}\boldsymbol{z} + \mathbf{b})$, where $\mathbf{W} \in \mathbb{R}^{k \times k}$ is a matrix of learned weights (restricted to the diagonal matrix for vector scaling, unrestricted for matrix scaling) and $\mathbf{b} \in \mathbb{R}^k$ is a vector of learned biases. Vector scaling is similar to temperature scaling, but instead of a single scaling parameter, a scaling parameter is learned separately for each class and an additional bias is also learned. For matrix scaling, each logit becomes a linear combination of other logits. Matrix scaling gives better results if it is trained with off-diagonal and intercept regularization (ODIR) to avoid overfitting due to the vast number of parameters [19]. The ODIR method proposed by Kull et al. [19] adds to the training loss the term $\lambda(\frac{1}{m(m-1)} \sum_{i \neq j} w_{ij}^2) + \mu(\frac{1}{m} \sum_j b_j^2)$, where λ and μ are the hyperparameters of the method. The first term of ODIR keeps the off-diagonal weights in $\mathbf{W}$ from growing too large and the second term adds loss for large biases in the bias vector.

2.5.4 Dirichlet calibration

Dirichlet calibration is a method proposed by Kull et al. [19] that is quite similar to matrix scaling, except it does not work on the logits of a neural network but rather on the actual predicted probabilities $\hat{\boldsymbol{p}}$ of a classifier. With Dirichlet calibration, the calibrated probabilities are obtained by $\hat{\boldsymbol{c}}(\hat{\boldsymbol{p}}) = \boldsymbol{\sigma}(\mathbf{W} \ln \hat{\boldsymbol{p}} + \mathbf{b})$, where $\mathbf{W} \in \mathbb{R}^{k \times k}$ is a matrix of learned weights, $\mathbf{b} \in \mathbb{R}^k$ a vector of learned biases, and $\ln$ is a vectorized natural-logarithm function. Similar to matrix scaling, Dirichlet calibration is also trained with ODIR to prevent overfitting.

2.5.5 Intra order-preserving functions

Rahimi et al. [38] proposed to use the family of intra order-preserving (IOP) functions to learn a calibration map on the logits of a neural network. An IOP function $\hat{\boldsymbol{c}} : \mathbb{R}^m \to \mathbb{R}^m$ is a vector-valued function where the order of the sorted output components is the same as the order of sorted input components, that is $argsort\, \hat{\boldsymbol{c}}(\boldsymbol{z}) = argsort\, \boldsymbol{z}$. The authors proposed to use IOP functions as they preserve the accuracy of the classifier and have larger expressive power than the scaling methods proposed by Guo et al. [8] or Dirichlet calibration. The IOP function family preserves classifier accuracy after calibration as the largest probability in the classifier output still remains the largest probability after calibration thanks to the order-preserving property. The IOP function family is more expressive than matrix scaling or Dirichlet calibration as IOP functions can learn non-linear transformations while matrix scaling and Dirichlet calibration are limited to linear transformations. Rahimi et al. [38] also found in their experiments

that in practice a *diagonal* subfamily of IOP functions performs better than the family of all IOP functions as the functions in the subfamily need less parameters and are easier to learn from the small calibration data set. An IOP function $\hat{\boldsymbol{c}}$ is a diagonal function if $\hat{\boldsymbol{c}}(\boldsymbol{z}) = (\hat{c}(z_1), \ldots, \hat{c}(z_m))$, where $\hat{c} : \mathbb{R} \to \mathbb{R}$ is a continuous and increasing function. A diagonal IOP function is symmetrical for all classes and produces output where the different class logits do not interact with each other in $\boldsymbol{c}$. It can be noted that temperature scaling uses a special case of diagonal IOP functions: it uses linear diagonal IOP functions where the bias term equals 0. Rahimi et al. [38] implemented the IOP post-hoc calibration method as a neural network with two fully connected hidden layers with order-preserving constraints. Their implementation has two hyperparameters: the number of neurons in the first and the second hidden layer.

In Figure 5 an illustrative example of several post-hoc calibration methods is shown. The calibration data set (validation data set) has been generated synthetically: the exact details of this synthetic calibration task are fully described in Section 4.2. The calibration data set is depicted in Figure 5a with the color depicting the label of the data point. The true calibration function $\boldsymbol{c}(\hat{\boldsymbol{p}}) = \mathbb{E}\left[\boldsymbol{Y}|\hat{\boldsymbol{P}} = \hat{\boldsymbol{p}}\right]$ is depicted in Figure 5b, where both the arrows and background colors depict the transformation of the function. The arrow drawn at $\hat{\boldsymbol{p}}$ is defined as $\boldsymbol{c}(\hat{\boldsymbol{p}}) - \hat{\boldsymbol{p}} = -CE(\hat{\boldsymbol{p}})$: it is the calibrating transformation or equivalently the negative calibration error. The background color at $\hat{\boldsymbol{p}}$ is drawn using the RGB color model with value $(0.5, 0.5, 0.5) + \lambda \cdot CE(\hat{\boldsymbol{p}})$, where λ defines the color intensity ($\lambda = 3.5$ for these figures). The value of each color channel is in a linear relation with the calibration error value for one of the classes: red for class 1, green for class 2, and blue for class 3. The gray baseline color $(0.5, 0.5, 0.5)$ shows no calibration transformation applied. Note that $(0.5, 0.5, 0.5)$ is defined as the baseline RGB value not $(0.0, 0.0, 0.0)$ as the calibration error vector can contain both negative and positive values. The goal of the calibration task is to estimate the true calibration function from the 5000 data points in the calibration data set (classifier predictions and labels). Note that in any real-world task, the true calibration function is never known — it is a magical function that would completely solve the problem of calibration. However, this is a synthetically generated example. Calibration maps are shown for histogram binning applied with 10 equal width bins in Figure 5c, isotonic regression in Figure 5d, beta calibration in Figure 5e, temperature scaling in Figure 5f, vector scaling in Figure 5g, Dirichlet calibration in Figure 5h and the intra order-preserving functions in Figure 5i. The binary methods have been applied in a one-vs-rest style. Note how the learned calibration map is not smooth for histogram binning and isotonic calibration as they both learn a piecewise constant map in the binary case. For histogram binning, the boundaries of the 10 bins can be clearly seen. Of the one-vs-rest methods, beta calibration seems to perform the best: the calibrating arrows of the true calibration map seem more similar to the arrows of beta calibration than to the arrows of histogram binning or isotonic

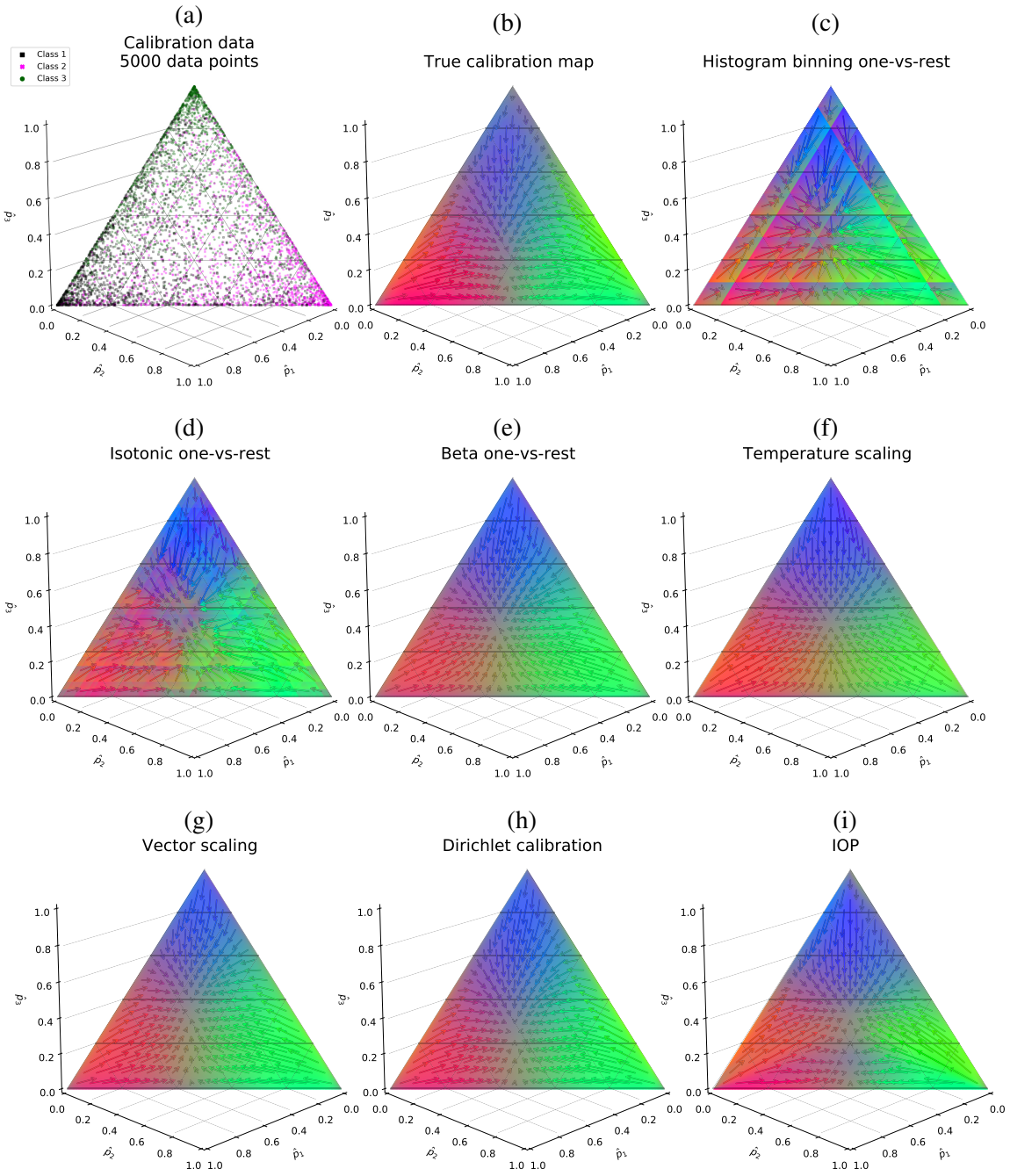

Figure 5. (a) A synthetically generated calibration data set of 5000 data points; (b) its true calibration map; (c-i) and the calibration maps learned by different post-hoc calibration methods.

regression. Temperature scaling learns a symmetric transformation for each class as the temperature parameter is shared across classes; overall, the temperature scaling family does not seem suitable for this particular synthetic example. Vector scaling is more expressive than temperature scaling and performs well. Dirichlet calibration captures the true map very well. Intra order-preserving functions learn a decent calibration map, capturing the key elements of the true calibration map. Section 4.2 includes a numerical comparison according to several evaluation metrics for some of the calibration maps in Figure 5.

2.6 Decision calibration

Another framework in which to view multi-class classifier calibration was introduced in a paper by Zhao et al. [46]. The authors argue that many machine learning systems operate in a decision-making scenario, where the only important differences in classifier predictions are the ones that lead to different decisions. Therefore, in a decision-making setting, it is not necessary to achieve the strict notion of multi-class calibration for the predictions, but rather predictions that are indistinguishable from calibrated predictions [46].

Decision-making scenario. In a decision-making scenario, the final output of a machine learning system is not a vector of predictions $\hat{\boldsymbol{p}}$, but rather some decision D chosen from a fixed set of available decisions $\mathcal{D}$ based on the predictions $\hat{\boldsymbol{p}}$. For example, the set of available decisions $\mathcal{D}$ could simply consist of the class labels $\mathcal{D} = \{\text{class } 1, \ldots, \text{class } m\}$, producing a classical classification task. However, as illustrated in the example in the next paragraph, more interesting sets could be defined as well. To go from a prediction $\hat{\boldsymbol{p}}$ to a decision D, a decision rule $\delta : \Delta^m \to \mathcal{D}$ is applied to the prediction $D = \delta(\hat{\boldsymbol{p}})$. The best decision rule is the Bayes decision rule that is determined by a loss function (cost matrix) $l : \mathcal{Y} \times \mathcal{D} \to \mathbb{R}$ that associates a cost to every combination of a decision and a label [46]. The Bayes decision rule chooses the decision that minimizes the expected loss. To be able to accurately choose the best decision that minimizes the expected loss, the prediction vector is assumed to be close to calibrated. If the prediction vector accurately represents the probability distribution of the label values, then the expected loss of every decision can be calculated with the loss function and the prediction vector.

Example scenario. As an example of a decision-making scenario, suppose a machine learning system deployed in a hospital that employs a classifier to predict the type of skin lesion depicted on a picture of a patient's skin. The classifier has four classes in the prediction task: one benign lesion class and three different cancerous lesion classes. However, the machine learning system has only two possible decisions in the final

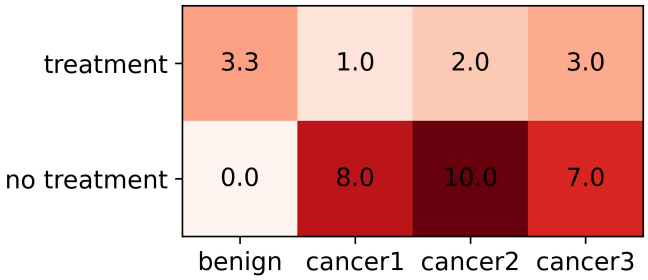

Figure 6. Example of a loss function (cost matrix). Adapted from Zhao et al. [46].

output: $\mathcal{D} = \{\textit{treatment}, \textit{no treatment}\}$. The hospital also has a loss function depicted in Figure 6 that associates a cost to every combination of a decision and a label. For example, according to the loss function, deciding *no treatment* when the true label of the image was a cancerous lesion belonging to class 4 has a high cost of 7; while deciding *no treatment* for the true label of a benign lesion has a low cost of 0. Whether the machine learning system opts for *treatment* or *no treatment* is uniquely determined by the prediction value and the hospital's loss function. To have an example of an optimal choice made by a Bayes decision rule, consider the loss function in Figure 6 and a model prediction $\hat{\boldsymbol{p}} = (0.60, 0.25, 0.05, 0.10)$. The expected loss for *treatment* is $0.60 \cdot 3.3 + 0.25 \cdot 1.0 + 0.05 \cdot 2.0 + 0.10 \cdot 3.0 = 2.63$; the expected loss for *no treatment* is $0.60 \cdot 0.0 + 0.25 \cdot 8.0 + 0.05 \cdot 10.0 + 0.10 \cdot 7.0 = 3.2$. As $2.63 < 3.2$, the Bayes decision rule would opt for *treatment*.

Definition of decision calibration. Given the decision making background, Zhao et al. [46] define decision calibration as follows: for some set of loss functions $\mathcal{L} = \{l_1, l_2, \dots\}$, a classifier $\boldsymbol{f}$ is considered to be $\mathcal{L}$-decision calibrated if for any $l \in \mathcal{L}$ and corresponding Bayes decision rule δ

$$\mathbb{E}_{\hat{\boldsymbol{P}}}\mathbb{E}_{\hat{\boldsymbol{Y}} \sim \hat{\boldsymbol{P}}}[l(\hat{\boldsymbol{Y}}, \delta(\hat{\boldsymbol{P}}))] = \mathbb{E}_{\hat{\boldsymbol{P}}}\mathbb{E}_{\boldsymbol{Y} \sim \boldsymbol{P}}[l(\boldsymbol{Y}, \delta(\hat{\boldsymbol{P}}))], \tag{1}$$

where $\boldsymbol{P} = \mathbb{E}[\boldsymbol{Y}|\boldsymbol{X}]$ is a random variable denoting the true label distribution corresponding to input features $\boldsymbol{X}$. The left-hand side of the equation is the expected loss if the labels were drawn from the predicted probability distributions. The right-hand side of the equation is the expected true loss. The definition provided by Zhao et al. [46] states that a classifier is considered to be decision calibrated according to some loss function if the expected loss is the same no matter whether the labels are sampled from the predictions the classifier outputs or from the true label distribution. Note that the definition only makes guarantees about the expectation of loss over all data points, not for any individual data point. The decision calibration definition encapsulates also the definitions of multi-class, classwise, and confidence calibration given in Section 2.2.

For example, Zhao et al. [46] show that a classifier being multi-class calibrated is the same as if it were decision calibrated with respect to the set of all possible loss functions $\mathcal{L} = \{l : \mathcal{Y} \times \mathcal{D} \to \mathbb{R}\}$.

Average loss gap. To be able to empirically compare calibration methods according to the decision calibration definition given in Equation 1, Zhao et al. [46] calculate the average loss gap over a set of randomly generated loss functions. In their work, they consider 500 random loss functions with 2 decisions and the costs generated from the standard normal distribution Normal$(0, 1)$. The loss gap is calculated for every generated loss function as

$$\text{loss gap} = \left| \mathbb{E}_{\hat{\boldsymbol{P}}} \mathbb{E}_{\hat{\boldsymbol{Y}} \sim \hat{\boldsymbol{P}}} [l(\hat{\boldsymbol{Y}}, \delta(\hat{\boldsymbol{P}}))] - \mathbb{E}_{\hat{\boldsymbol{P}}} \mathbb{E}_{\boldsymbol{Y} \sim \boldsymbol{P}} [l(\boldsymbol{Y}, \delta(\hat{\boldsymbol{P}}))] \right|.$$

The expectations over classifier predictions $\mathbb{E}_{\hat{\boldsymbol{P}}}$ are replaced with the empirical averages on the test set predictions; the expectation over the true labels $\mathbb{E}_{\boldsymbol{Y} \sim \boldsymbol{P}}$ is replaced with the empirical average on the test set labels; and the average loss gap over 500 loss functions is reported.

Post-hoc decision calibration method. In addition to the encapsulating framework of decision calibration, Zhao et al. [46] also propose a non-parametric post-hoc calibration algorithm to achieve decision calibration with respect to the set of all loss functions with k decisions $\mathcal{L}_k = \{l : \mathcal{Y} \times \mathcal{D} \to \mathbb{R}, |\mathcal{D}| = k\}$. The proposed algorithm has theoretical guarantees to achieve decision calibration up to a selected precision level given enough data. It is able to do this thanks to the observation that the Bayes decision rule δ_l for any $l \in \mathcal{L}_k$ partitions the probability simplex by linear classification boundaries. The algorithm works iteratively by finding in each iteration the partition b of the probability simplex into k parts with linear boundaries where calibration is violated the most and improving the worst split.

Before providing the full algorithm description, consider the example in Figure 7 which illustrates the algorithm result after 1, 2, and 10 iterations trained to achieve decision calibration with respect to all loss functions with 2 decisions on the same synthetic data set as in Figure 5. At every iteration, the algorithm partitions the probability simplex into 2 with a linear boundary. On each side of the partition, the algorithm learns an adjustment equal to the average difference between the label and prediction values. The place where to partition is determined by the adjustment size needed. The partition with the largest needed adjustments is determined as the best one. Put more intuitively but a bit simplified, the algorithm searches for a partition, where in each part of the partition, the average arrow on the true calibration map it is trying to learn is as long as possible. Applying the partition found after the first iteration in Figure 7a on the true calibration map in Figure 5b, the average arrow in each part of the partition would indeed be long. Another point to notice from Figure 7a, is that the found partition is

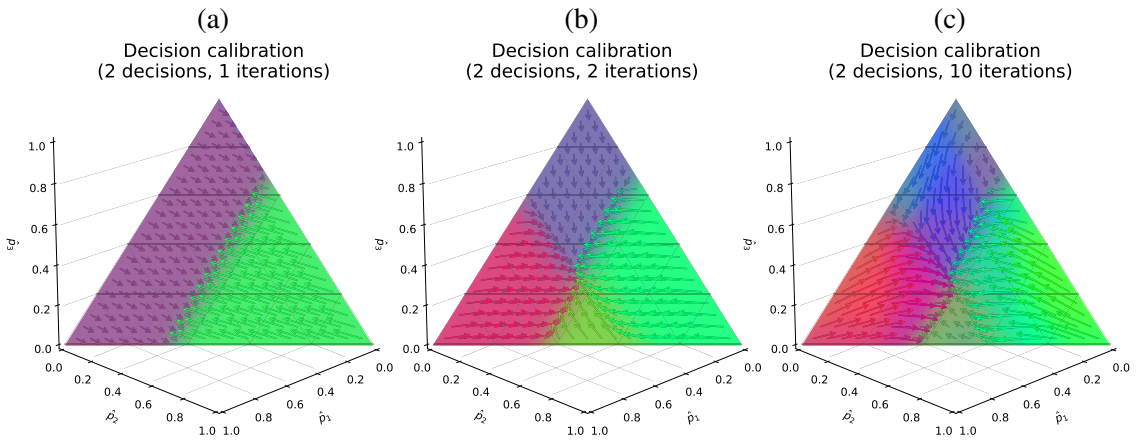

Figure 7. Calibration maps produced by decision calibration algorithm after (a) 1, (b) 2, and (c) 10 iterations. Trained to achieve decision calibration with respect to all loss functions with 2 decisions on the same synthetic data set as in Figure 5.

not exactly defined by a linear boundary but by a more fuzzy one. This is just due to implementational limitations as a fuzzy boundary allows for differentiation when looking for the best partition while a strictly linear one does not.

The slightly simplified algorithm description is given in Algorithm 1. The partition b is an indicator function that describes a specific partition of the probability simplex into k parts. Given a prediction vector $\hat{\boldsymbol{p}}$ and the index of the i-th part, $b(\hat{\boldsymbol{p}}, i) = 1$ if the vector $\hat{\boldsymbol{p}}$ belongs to part i and $b(\hat{\boldsymbol{p}}, i) = 0$ if it does not. The empirical expectations $\hat{\mathbb{E}}$ are defined as averages on the calibration data set.

Algorithm 1: Calibration algorithm to achieve $\mathcal{L}_k$ decision calibration [46].

Input: Current classifier function $\boldsymbol{f}$, tolerance ϵ.
Output: Decision calibrated classifier function $\boldsymbol{f}^{(t)}$.

$v = \infty; t = 0; \boldsymbol{f}^{(t)} = \boldsymbol{f}$
while $v \geq \epsilon$
  $t = t + 1$
  Find a partition b with linear boundaries into k parts that maximizes the calibration violation

$$v = \sum_{i=1}^{k} ||\hat{\mathbb{E}}[(\boldsymbol{Y} - \boldsymbol{f}^{(t-1)}(\boldsymbol{X}))b(\boldsymbol{f}^{(t-1)}(\boldsymbol{X}), i)]||$$

  Compute adjustments for each partition $i = 1, \ldots, k$

$$\boldsymbol{a}_i = \hat{\mathbb{E}}[(\boldsymbol{Y} - \boldsymbol{f}^{(t-1)}(\boldsymbol{X}))b(\boldsymbol{f}^{(t-1)}(\boldsymbol{X}), i)]/\hat{\mathbb{E}}[b(\boldsymbol{f}^{(t-1)}(\boldsymbol{X}), i)]$$

  Update the classifier function

$$\boldsymbol{f}^{(t)} : \boldsymbol{x} \to \boldsymbol{f}^{(t-1)}(\boldsymbol{x}) + \sum_{i=1}^{k} b(\boldsymbol{f}^{(t-1)}(\boldsymbol{x}), i) \cdot \boldsymbol{a}_i$$

end
return $\boldsymbol{f}^{(t)}$

3 Contributions

The contributions of this thesis to the field of multi-class calibration can be split into two:

1. the thesis formulates two assumptions about the true calibration map that have been previously used but not clearly stated in the calibration literature;

2. based on the more reasonable assumption of the two, the thesis proposes three post-hoc calibration methods: random forest calibration, k-nearest-neighbors calibration, and kernel calibration.

3.1 Proposed assumptions

Before introducing the proposed assumptions, a small introduction is needed. According to the definition of multi-class calibration, a prediction vector $\hat{\boldsymbol{p}}$ is considered calibrated if it is equal to $\mathbb{E}\left[\boldsymbol{Y}|\hat{\boldsymbol{P}} = \hat{\boldsymbol{p}}\right]$ or equivalently equal to $\hat{\boldsymbol{p}} - CE(\hat{\boldsymbol{p}})$. Therefore, to calibrate prediction $\hat{\boldsymbol{p}}$, all that is needed is to be able to estimate the true label distribution $\mathbb{E}\left[\boldsymbol{Y}|\hat{\boldsymbol{P}} = \hat{\boldsymbol{p}}\right]$ or the calibration error vector $CE(\hat{\boldsymbol{p}})$. Given a data point with predicted output $\hat{\boldsymbol{p}}$ and label $\boldsymbol{y}$, the value of $\mathbb{E}\left[\boldsymbol{Y}|\hat{\boldsymbol{P}} = \hat{\boldsymbol{p}}\right]$ can be estimated with just $\boldsymbol{y}$ as it is a realization of the random variable $\boldsymbol{Y}$ under the condition that $\hat{\boldsymbol{P}} = \hat{\boldsymbol{p}}$. Similarly, the value of $CE(\hat{\boldsymbol{p}}) = \hat{\boldsymbol{p}} - \mathbb{E}\left[\boldsymbol{Y}|\hat{\boldsymbol{P}} = \hat{\boldsymbol{p}}\right]$ can be estimated with just $\widehat{CE}(\hat{\boldsymbol{p}}) = \hat{\boldsymbol{p}} - \boldsymbol{y}$. Even though both of the estimates are unbiased, they have too high variance to be useful alone in practice: they are based on a sample with just one element. Additionally, they are only available on the calibration data set where the label values $\boldsymbol{y}$ are known: to calibrate a prediction on the test set where the true label value is not available, these estimates can not be constructed. These two problems can be solved if one would make additional assumptions.

Assumption of equal calibration errors. This thesis proposes to assume that the calibration error $CE(\hat{\boldsymbol{p}}) = \hat{\boldsymbol{p}} - \mathbb{E}\left[\boldsymbol{Y}|\hat{\boldsymbol{P}} = \hat{\boldsymbol{p}}\right]$ is approximately equal in a close neighborhood on the probability simplex:

$$\text{if } \hat{\boldsymbol{p}}_1 \approx \hat{\boldsymbol{p}}_2 \text{ then } CE(\hat{\boldsymbol{p}}_1) \approx CE(\hat{\boldsymbol{p}}_2).$$

Given a prediction $\hat{\boldsymbol{p}}$, this assumption allows to construct a decent estimate for the calibration error of $\hat{\boldsymbol{p}}$. To begin, the close neighborhood of $\hat{\boldsymbol{p}}$ in the calibration data set has to be found. Suppose that the close neighborhood contains data points $\hat{\boldsymbol{p}}_1, \ldots, \hat{\boldsymbol{p}}_k$ with labels $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_k$. For every found data point $\hat{\boldsymbol{p}}_i$ in the neighborhood an unbiased but a highly varying estimate of its calibration error $\widehat{CE}(\hat{\boldsymbol{p}}_i) = \hat{\boldsymbol{p}}_i - \boldsymbol{y}_i$ is known. Thanks

to the assumption $CE(\hat{\boldsymbol{p}}) \approx CE(\hat{\boldsymbol{p}}_1) \approx \cdots \approx CE(\hat{\boldsymbol{p}}_k)$ one can then construct a decent estimate for the calibration error of the initial prediction $\hat{\boldsymbol{p}}$ by averaging across the estimates of its neighbors

$$\widehat{CE}_{neigh}(\hat{\boldsymbol{p}}) = \frac{1}{k} \sum_{i=1}^{k} \widehat{CE}(\hat{\boldsymbol{p}}_i).$$

The estimate remains unbiased as

$$\mathbb{E}\left[\widehat{CE}_{neigh}(\hat{\boldsymbol{p}})\right] = \frac{1}{k} \sum_{i=1}^{k} \mathbb{E}\left[\widehat{CE}(\hat{\boldsymbol{p}}_i)\right] = \frac{1}{k} \sum_{i=1}^{k} CE(\hat{\boldsymbol{p}}_i) \approx \frac{1}{k} \sum_{i=1}^{k} CE(\hat{\boldsymbol{p}}) = CE(\hat{\boldsymbol{p}}).$$

The variance of the estimate lowers approximately linearly as the number of neighbors increases

$$Var\left[\widehat{CE}_{neigh}(\hat{\boldsymbol{p}})\right] = Var\left[\frac{1}{k} \sum_{i=1}^{k} \widehat{CE}(\hat{\boldsymbol{p}}_i)\right] = \frac{\sum_{i=1}^{k} Var[\widehat{CE}(\hat{\boldsymbol{p}}_i)]}{k^2} \approx \frac{Var[\widehat{CE}(\hat{\boldsymbol{p}})]}{k}.$$

Given the estimate $\widehat{CE}_{neigh}(\hat{\boldsymbol{p}})$, a calibrated prediction can be constructed by subtracting it from the original prediction $\hat{\boldsymbol{c}}(\hat{\boldsymbol{p}}) = \hat{\boldsymbol{p}} - \widehat{CE}_{neigh}(\hat{\boldsymbol{p}})$.

Assumption of equal label distributions. A similar derivation for constructing calibrated predictions is possible if one would instead assume that the true label distribution $\mathbb{E}\left[\boldsymbol{Y}|\hat{\boldsymbol{P}} = \hat{\boldsymbol{p}}\right]$ is approximately equal in a close neighborhood:

$$\text{if } \hat{\boldsymbol{p}}_1 \approx \hat{\boldsymbol{p}}_2 \text{ then } \mathbb{E}\left[\boldsymbol{Y}|\hat{\boldsymbol{P}} = \hat{\boldsymbol{p}}_1\right] \approx \mathbb{E}\left[\boldsymbol{Y}|\hat{\boldsymbol{P}} = \hat{\boldsymbol{p}}_2\right].$$

With this assumption, the method would arrive at a calibrated prediction by using the average label value in a close neighborhood $\hat{\boldsymbol{c}}(\hat{\boldsymbol{p}}) = \frac{1}{k} \sum_{i=1}^{k} \boldsymbol{y}_i$.

Visualisation of the assumptions. To better understand the difference between the two assumptions, an illustration is given in Figure 8. On both of the subfigures, a close neighborhood of the 2-dimensional probability simplex is depicted with the corresponding assumption applied. If compared with the true calibration map in Figure 5b, the assumption of similar calibration errors in a close neighborhood is visually clearly superior: the calibration transformation arrows in Figure 5b tend to be parallel and point in the same direction, not to the same location. The assumption of similar calibration errors is preferred to the alternative in this thesis.

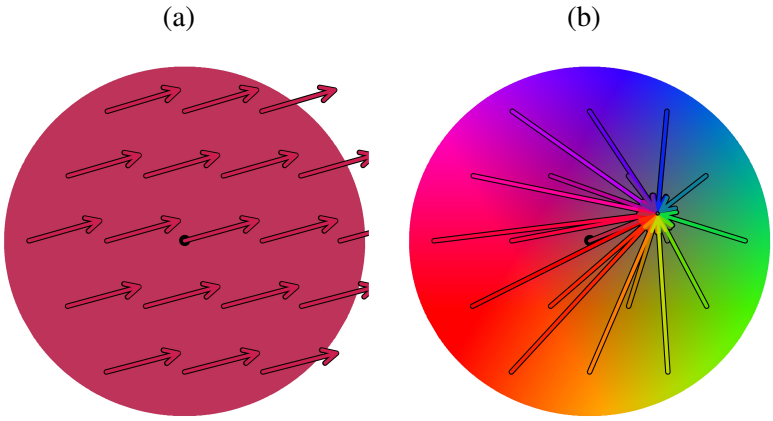

Figure 8. Different assumptions about the true calibration map. (a) Assuming calibration error to be similar in a close neighborhood (proposed methods). (b) Assuming true label distribution to be similar in a close neighborhood (histogram binning).

Relation to prior work. Neither of the assumptions is new. The assumption of equal calibration errors has been used in at least two algorithms that use the average difference between the label and prediction in a close neighborhood. First, the ECE calculation algorithm [31] defines the close neighborhood with bins and in each bin uses the difference between the average label and average prediction value to estimate the calibration error — this is exactly the same as the proposed assumption as $\frac{1}{k}\sum_{i=1}^{k}\hat{\boldsymbol{p}}_i - \frac{1}{k}\sum_{i=1}^{k}\boldsymbol{y}_i = \frac{1}{k}\sum_{i=1}^{k}(\hat{\boldsymbol{p}}_i - \boldsymbol{y}_i)$. Second, the recently proposed decision calibration algorithm [46] described in Algorithm 1 also uses the average difference between the predictions and labels in a close neighborhood. In the decision calibration algorithm, the close neighborhoods are defined in each iteration by the partition b.

The assumption of equal label distributions has also been used previously. It is the basis of the histogram binning method [42] where the close neighborhood is defined by the binning scheme; in each bin, the average label value is used to calibrate the predictions. The weighted average label value is also used in recent work by Popordanoska et al. [37] for a training time calibration method, not a post-hoc calibration method. In their work, all the neighbors of a prediction are assigned a weight with a kernel function; the weighted average of label values is then used to estimate the calibrated prediction. While not using the same assumption of equal label distributions, it assumes something similar: that the label distributions can be modeled by their neighbors, and the best way to model is defined by the weights assigned by the kernel function.

Even though both assumptions have been used in calibration literature, they have never been clearly stated. The algorithms for calculating ECE and for histogram binning have been proposed without clearly explained assumptions. For the decision calibration

algorithm, the use of the average difference between the label and prediction value was well justified: the algorithm was derived directly from the decision calibration definition [46]. However, the intuitive meaning of $\frac{1}{k}\sum_{i=1}^{k}(\hat{\boldsymbol{p}}_i - \boldsymbol{y}_i)$ was not explained to the reader in the original article.

Defining the close neighborhood. For both assumptions, two open questions remain:

1. how large should the close neighborhood be;
2. how should the close neighborhood be defined?

To answer the first question: the more neighbors taken, the less varying the estimators; however, the further away the neighbors are, the bolder the assumptions and the larger the bias from the assumption. Therefore, a sweet spot for the bias-variance tradeoff has to be found. This can be achieved if the used neighborhood scheme offers a hyperparameter defining the neighborhood size. The hyperparameter could then be optimized with cross-validation with Brier score or log-loss.

There is no clear correct answer to the second question. Histogram binning and the ECE algorithm define the close neighborhood with a binning scheme. However, the binning schemes are only defined for the binary case. The decision calibration algorithm defines the close neighborhoods by a linear partition b that splits the probability simplex such that the calibration error estimates would be maximized. Popordanoska et al. [37] define in their method the neighborhood through assigning weights with a kernel function. This thesis proposes to use three multi-dimensional neighborhood schemes and therefore offers three new post-hoc calibration methods: random calibration forests, k-nearest-neighbors calibration, and kernel calibration.

3.2 Random calibration forest

The first proposed post-hoc calibration method uses bins to define the close neighborhoods similar to the ECE algorithm and histogram binning. The difference being, that it extends binning beyond the binary case by taking inspiration from randomized decision trees. The proposed neighborhood scheme divides the multi-dimensional probability simplex into bins by picking a random class i and uniformly randomly a prediction value r in the range of this class probabilities, and splitting the simplex into two with the decision rule

$$d(\hat{\boldsymbol{p}}) = \begin{cases} \text{bin 1}, & \text{if } \hat{p}_i < r \\ \text{bin 2}, & \text{if } \hat{p}_i \geq r \end{cases}.$$

The splitting process is then recursively repeated (breadth-first) in both of the bins until the desired number of total bins is reached. Finally, in each bin, the assumption of equal calibration errors is used to calibrate the predictions belonging to it. In this thesis,

such a post-hoc calibration method is called a random calibration tree. In Figure 9a and Figure 9b, two examples of random calibration trees with 32 bins are shown on the same synthetic calibration task as in Figure 5. Note the large differences in bin sizes that appear due to the random nature of choosing the bin borders.

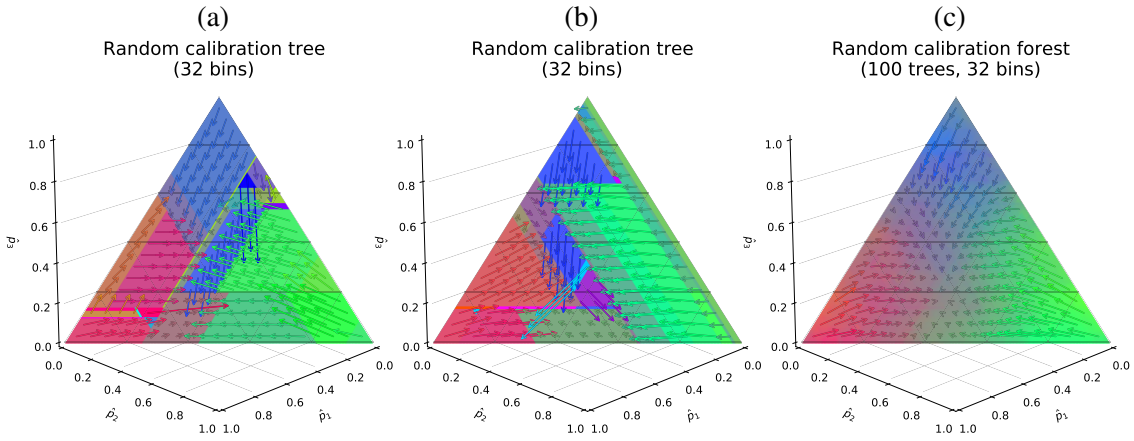

Figure 9. (a) (b) Two examples of random calibration trees with 32 bins; (c) 100 such trees combined to a random calibration forest. Trained on the same synthetic data set as in Figure 5.

For random calibration trees, the assumption of equal calibration errors becomes very bold in some constructed bins. In addition, the algorithm produces highly varying calibration maps with each re-run as the bins are chosen arbitrarily. A better post-hoc calibration method with less variance can be constructed by taking inspiration from decision forests and simply averaging across several random calibration trees producing a random calibration forest. In Figure 9c an example of a random calibration forest combining 100 32-bin random calibration trees is shown. For random calibration forests, the size of the close neighborhood is determined by the number of bins chosen: the more bins, the smaller they are on average.

3.3 KNN calibration

The second proposed method works similarly to the k-nearest-neighbors (KNN) algorithm: the close neighborhood of prediction $\hat{\boldsymbol{p}}$ is defined by the k nearest data points according to some distance function d. For any uncalibrated prediction $\hat{\boldsymbol{p}}$, the KNN post-hoc calibration method would

1. find the closest k-predictions on the calibration data set $\hat{\boldsymbol{p}}_1, \ldots, \hat{\boldsymbol{p}}_k$ according to distance function d;

2. estimate the calibration error by finding the average difference between the neighbors' prediction and label $\widehat{CE}_{knn}(\hat{\boldsymbol{p}}) = \frac{1}{k}\sum_{i=1}^{k}(\hat{\boldsymbol{p}}_i - \mathbf{y}_i) = \frac{1}{k}\sum_{i=1}^{k}\widehat{CE}(\hat{\boldsymbol{p}}_i)$;

3. subtract the calibration error from the uncalibrated prediction to calibrate it $\hat{\boldsymbol{c}}(\hat{\boldsymbol{p}}) = \hat{\boldsymbol{p}} - \widehat{CE}_{knn}(\hat{\boldsymbol{p}})$.

For the distance function d, two choices are considered in this thesis: Euclidean distance and Kullback-Leibler divergence. Euclidean distance is defined as

$$d_{euc}(\hat{\boldsymbol{p}}_1, \hat{\boldsymbol{p}}_2) = ||\hat{\boldsymbol{p}}_1 - \hat{\boldsymbol{p}}_2||_2$$

and while commonly used, it is not a good function for high dimensional spaces where all points are roughly uniformly distant from each other [1]. Kullback-Leibler divergence is a distance to measure the difference between two probability distributions: $d_{kl}(\hat{\boldsymbol{p}}_1, \hat{\boldsymbol{p}}_2)$ describes the level on surprise when the actual probability distribution is $\hat{\boldsymbol{p}}_1$ but it is modeled with $\hat{\boldsymbol{p}}_2$ [30]. It is defined as

$$d_{kl}(\hat{\boldsymbol{p}}_1, \hat{\boldsymbol{p}}_2) = \sum_{i=1}^{m} \hat{p}_{1i} \log\left(\frac{\hat{p}_{1i}}{\hat{p}_{2i}}\right),$$

where the term in the sum is considered 0 if $\hat{p}_{1i} = 0$ and ∞ if $\hat{p}_{1i} \neq 0$ and $\hat{p}_{2i} = 0$ [30]. It is important to note that while Kullback-Leibler divergence can be used to measure distance between to probability distributions, it is not a distance metric as it is asymmetric: $d_{kl}(\hat{\boldsymbol{p}}_1, \hat{\boldsymbol{p}}_2) \neq d_{kl}(\hat{\boldsymbol{p}}_2, \hat{\boldsymbol{p}}_1)$. To find the neighbors of $\hat{\boldsymbol{p}}$ on the calibration data set, $\hat{\boldsymbol{p}}$ is defined as the true probability distribution while the predictions on the

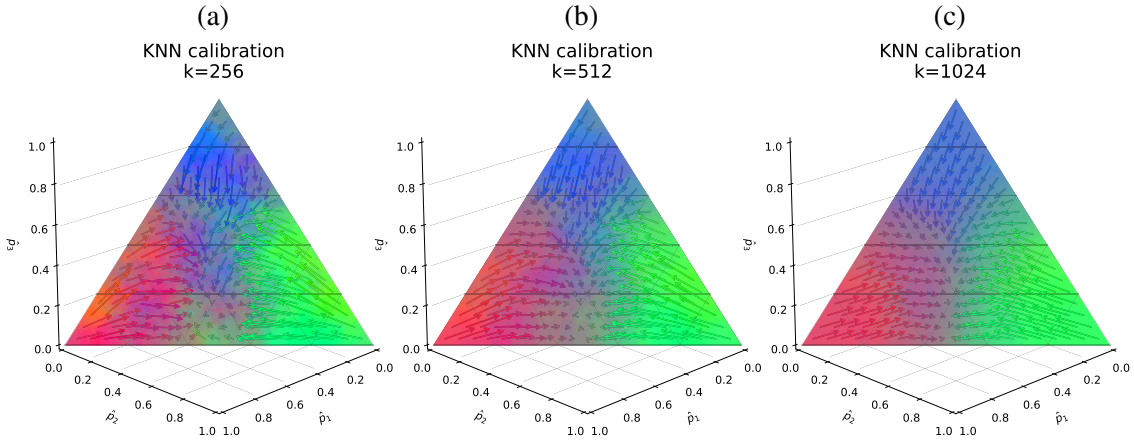

Figure 10. KNN calibration with Euclidean distance and (a) 256, (b) 512 and (c) 1024 neighbors. Trained on the same synthetic data set as in Figure 5.

calibration data set are considered the possible distributions to model $\hat{\boldsymbol{p}}$: the distance function is applied as $d_{kl}(\hat{\boldsymbol{p}}, \cdot)$.

Figure 10 shows an illustrative example of the calibration maps learned by KNN calibration with Euclidean distance and 256, 512, and 1024 neighbors on the same synthetic data set as in Figure 5. Note how smaller values of neighbors produce more noisy calibration maps.

3.4 Kernel calibration

The basis of random calibration forests and KNN calibration is the assumption that neighbors in close proximity have equal calibration error. One could argue that it would be more reasonable to assume that the calibration error of neighbors is not equal in a close proximity, but rather it is in a relation to the distance from each other and the optimal way to model that relation is with some weighting function. A smooth dependency between the calibration errors could be modeled with a kernel density function. Given a calibration data set consisting of predictions $\hat{\boldsymbol{p}}_1, \ldots, \hat{\boldsymbol{p}}_n$ and corresponding labels $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n$, the calibration error of prediction $\hat{\boldsymbol{p}}$ could be estimated by

$$\begin{aligned}\widehat{CE}_{kernel}(\hat{\boldsymbol{p}}) &= \frac{1}{\sum_{i=1}^{n} K(\hat{\boldsymbol{p}}, \hat{\boldsymbol{p}}_i)} \sum_{i=1}^{n} K(\hat{\boldsymbol{p}}, \hat{\boldsymbol{p}}_i)(\hat{\boldsymbol{p}}_i - \mathbf{y}_i) \\ &= \frac{1}{\sum_{i=1}^{n} K(\hat{\boldsymbol{p}}, \hat{\boldsymbol{p}}_i)} \sum_{i=1}^{n} K(\hat{\boldsymbol{p}}, \hat{\boldsymbol{p}}_i)\widehat{CE}(\hat{\boldsymbol{p}}_i),\end{aligned}$$

where K is the kernel function and the fraction $\frac{1}{\sum_{i=1}^{n} K(\hat{\boldsymbol{p}}, \hat{\boldsymbol{p}}_i)}$ is used to normalize the sum of weights assigned by the kernel function to 1. A similar assumption has been used by Popordanoska et al. [37] for a train time calibration method, not a post-hoc method, that models the label distributions with weights assigned by a Dirichlet kernel function [35].

In this thesis, the radial basis function (RBF) kernel [30] and the Dirichlet kernel [35] are considered but other kernels could be used as well. RBF kernel is defined to be

$$K_{RBF}(\hat{\boldsymbol{p}}_1, \hat{\boldsymbol{p}}_2) = \exp(-\gamma ||\hat{\boldsymbol{p}}_1 - \hat{\boldsymbol{p}}_2||_2^2),$$

where γ is the bandwidth parameter of the RBF kernel defining the size of the neighborhood [30]. Dirichlet kernel is defined to be the probability density function of the Dirichlet distribution

$$K_{DIR}(\hat{\boldsymbol{p}}_1, \hat{\boldsymbol{p}}_2) = \frac{\Gamma(\sum_{i=1}^{m} \alpha_i)}{\prod_{i=1}^{m} \Gamma(\alpha_i)} \prod_{i=1}^{m} \hat{p}_{2i}^{\alpha_i - 1},$$

where $\alpha_i = \hat{p}_{1i}/h + 1$ is the i-th distribution parameter, and h the kernel bandwidth defining the size of the neighborhood [35].

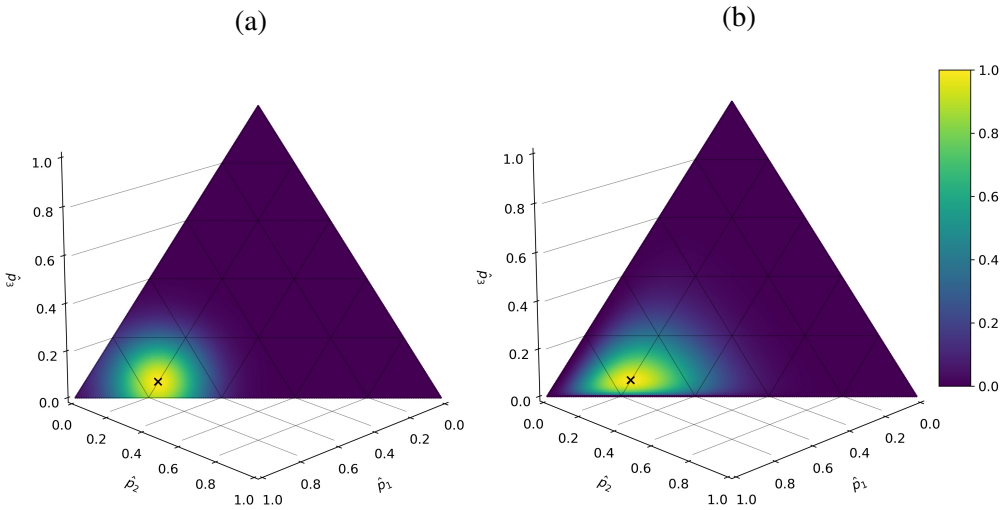

Figure 11. (a) $K_{RBF}(\hat{\boldsymbol{p}}_1, \hat{\boldsymbol{p}}_2)$ values for $\hat{\boldsymbol{p}}_1 = (0.75, 0.2, 0.05)$ with kernel parameter $\gamma = 30$. (b) $K_{DIR}(\hat{\boldsymbol{p}}_1, \hat{\boldsymbol{p}}_2)$ values for $\hat{\boldsymbol{p}}_1 = (0.75, 0.2, 0.05)$ with kernel parameter $h = 0.1$ (values have been scaled between 0 and 1).

In Figure 11 are examples of the weights assigned by the RBF kernel and the Dirichlet kernel on a 2-dimensional probability simplex. It can be seen that the RBF kernel is symmetrical in all directions while the Dirichlet kernel is not.

Figure 12 shows an illustrative example of the calibration maps learned by kernel calibration with Dirichlet kernel and kernel parameters 0.01, 0.1, and 1 on the same synthetic data set as in Figure 5. Note how a parameter value too small in Figure 12a

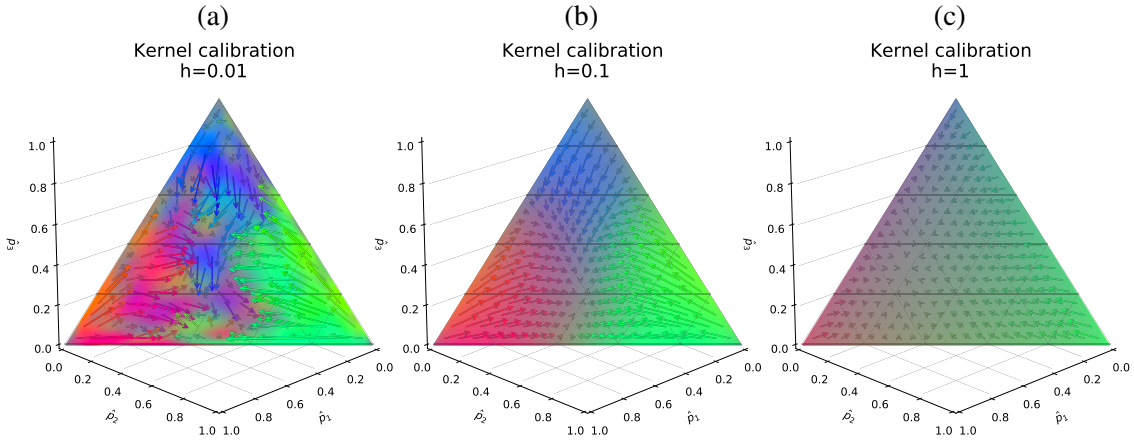

Figure 12. Kernel calibration with Dirichlet kernel and parameter values (a) 0.01, (b) 0.1, (c) 1. Trained on the same synthetic data set as in Figure 5.

learns a very noisy calibration map similarly to KNN calibration with too few neighbors as it takes into consideration only the very close neighbors. A well-chosen parameter value in Figure 12b produces a calibration map very similar to the true calibration map depicted in Figure 5. A parameter value too large in Figure 12c, however, produces an uninformative calibration map as it starts considering every data point in the calibration data set with approximately equal weight; essentially assuming that the calibration error is approximately equal for all data.

3.5 Additional details

The following paragraphs address some additional shared details for the proposed methods.

Cropping. A flaw with all the proposed non-parametric methods is that they might produce calibrated predictions that are not on the probability simplex. In some cases when the original prediction is close to the border of the simplex and the estimated calibration error points out of the simplex, nothing stops the calibrated prediction from leaving the probability simplex. This can be seen happening in Figure 9b in the top where no cropping has been used and one of the arrows goes slightly out of the simplex. To combat this, all the methods have to be accompanied by some cropping function that prevents the predictions from leaving the simplex.

Algorithm 2: Cropping function.

Input: Prediction value $\hat{\boldsymbol{p}} = (\hat{p}_1, \ldots, \hat{p}_m)$ that may not be on the probability simplex, small value ϵ.

Output: Cropped prediction $\hat{\boldsymbol{p}}'$ that is on the probability simplex.

1 $\hat{\boldsymbol{p}}' = (\min\{\max\{\hat{p}_1, \epsilon\}, 1 - \epsilon\}, \ldots, \min\{\max\{\hat{p}_m, \epsilon\}, 1 - \epsilon\})$
2 $\hat{\boldsymbol{p}}' = \hat{\boldsymbol{p}}' / ||\hat{\boldsymbol{p}}'||_1$
3 **return** $\hat{\boldsymbol{p}}'$

The cropping function used is described in Algorithm 2. The function first sets in line 1 every value smaller than 0 to ϵ and every value larger than 1 to $1 - \epsilon$. The predictions are not cropped to exactly 0 nor to exactly 1 to avoid producing infinitely large log-loss. In the experiments, ϵ was set to 10^{-4} or 10^{-6} correspondingly for data sets with 10 and 100 classes. Cropping some of the values in $\hat{\boldsymbol{p}}$ produces a vector that no longer sums to one and can not be interpreted as probabilities. Therefore, in line 2 of the algorithm, the output is normalized to sum to one. This is done by dividing the vector by its sum.

Composition with parametric methods. The authors of the decision calibration algorithm noticed that their proposed non-parametric post-hoc calibration method works better if it is applied in composition with temperature scaling [46]. First, temperature scaling learns the calibration map on the calibration data set and then their method fine-tunes the result of temperature scaling using the temperature scaled calibration data. The benefit of composing parametric and non-parametric calibration methods was also shown by Zhang et al. [45] who noted that isotonic regression applied in a one-versus-rest manner works better if it is applied on top of temperature scaling. Composing two post-hoc calibration methods is illustrated in Figure 13. A similar observation is true for the proposed non-parametric methods in this thesis as well. The experiments on real data in Section 4.3 show that the existing parametric post-hoc calibration methods significantly outperform the proposed non-parametric methods on the selected data sets. However, improvements can be seen when the proposed methods are applied in composition with parametric methods.

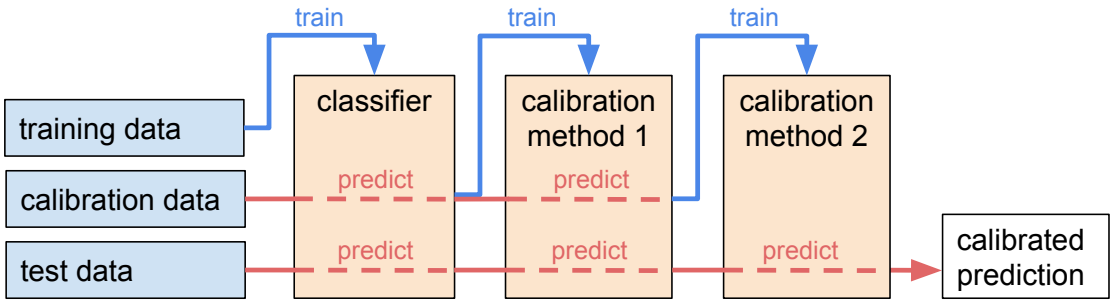

Figure 13. Schematic visualisation of using a composition of post-hoc calibration methods.

4 Experiments

The experiments' section consists of two parts[2]:

- a small experiment on synthetically generated data. The goal of this experiment is to illustrate and give visual intuition about the different post-hoc calibration methods.
- larger experiments on two real data sets and three convolutional neural network classifiers. The goal of these experiments is to compare the proposed post-hoc calibration methods with their competitors and see if they can offer improvement over the state-of-the-art.

4.1 Evaluation metrics

The methods are compared according to several metrics:

- classwise ECE and confidence ECE — as they are direct measures of calibration (see Section 2.4.1 for details). For synthetic experiments, the ECE measures are replaced with the true classwise calibration error (classwise CE) and true confidence calibration error (confidence CE) as the true calibration function is known (see Section 2.3 for details);
- Brier score and log-loss — as their minimum is defined by a multi-class calibrated classifier (see Section 2.4.2 for details);
- accuracy — to ensure the methods do not worsen the model's classification performance;
- the average loss gap over a set of randomly generated loss functions — to evaluate the goodness according to the decision-making framework (see Section 2.6 for details).

The true calibration errors and ECE values are calculated with absolute distance; the ECE values are calculated with 15 equal-sized bins; ECE values are also multiplied by 100 similarly to Guo et al. [8] so they can be interpreted as percentages of the theoretical maximum absolute ECE value of 1.

The average loss gap is calculated similarly to the experiments of Zhao et al. [46]: 500 random loss functions with 2 decisions are considered with weights sampled from the standard normal distribution Normal$(0, 1)$; the reported result is the average loss gap over the 500 loss functions. For the synthetic experiment, the average loss gap is calculated more precisely: the expectation over the true labels $\mathbb{E}_{\boldsymbol{Y} \sim \boldsymbol{P}}$ is not replaced with the empirical average on the test set labels as the true calibration map is known.

[2]The code for the experiments is available at https://github.com/kaspar98/msc-knn-cal.

4.2 Synthetic data experiment

The goal of the synthetic experiment is to give visual intuition about the post-hoc calibration methods and their learned calibration maps. The data are generated synthetically as this allows to define the magical true calibration function $\boldsymbol{c}(\hat{\boldsymbol{p}}) = \mathbb{E}[\boldsymbol{Y}|\hat{\boldsymbol{P}} = \hat{\boldsymbol{p}}]$, which is not available in any real-world data set: if it would be, there would be no need for calibration methods. Having access to the true calibration function allows for comparison between the results produced by the different post-hoc calibration methods and the optimal result of the true calibration function. The following subsections first describe the synthetic data generation, then discuss the details of methods under comparison, and finally compare the results.

4.2.1 Data generation

For the experiment, a 3-class calibration data set consisting of 5000 prediction-label pairs and a test data set consisting of 100000 prediction-label pairs were generated. Classifier predictions $\hat{\boldsymbol{p}}$ were sampled from a 3-class Dirichlet distribution with parameters $[0.5, 0.5, 0.5]$; the label distributions were calculated by applying the chosen true calibration function $\boldsymbol{c}(\hat{\boldsymbol{p}}) = \mathbb{E}[\boldsymbol{Y}|\hat{\boldsymbol{P}} = \hat{\boldsymbol{p}}]$ on the predictions; the labels $\boldsymbol{y}$ were sampled from $\boldsymbol{c}(\hat{\boldsymbol{p}})$. The chosen true calibration function $\boldsymbol{c}$ is depicted in Figure 14 on the right; the generated calibration data set is depicted on the left. Note that the synthetic data set is the same as in Figure 5.

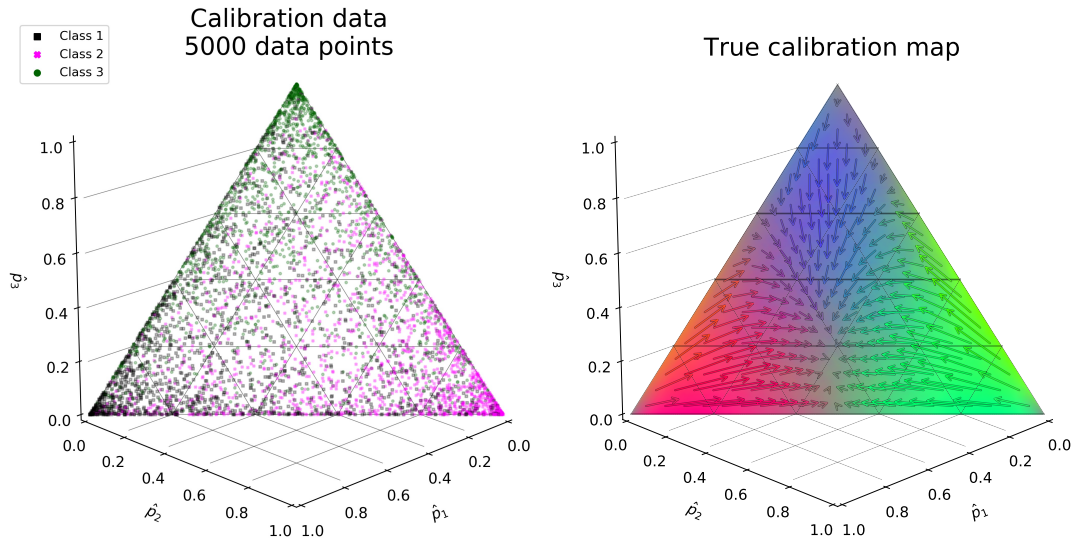

Figure 14. The synthetically generated calibration data set and the true calibration map used to calculate the true label distribution in the data generation process. The same two illustrations are also in Figure 5.

For the purpose of generating a synthetic data set, any desired shape could be chosen as the true calibration function $\boldsymbol{c}(\hat{\boldsymbol{p}}) = \mathbb{E}[\boldsymbol{Y}|\hat{\boldsymbol{P}} = \hat{\boldsymbol{p}}]$. For this experiment, it was important that the chosen shape would fulfill the following properties:

- As the main purpose of this experiment is to illustrate the different post-hoc calibration methods trying to imitate the true calibration function, the chosen shape should be complex enough to be able to differentiate between the different methods; yet not too complex to be unable to visualize clearly.
- To increase the credibility of the experiment, the chosen true calibration function should be realistic to some extent.

Therefore, the shape in Figure 14 on the right was not defined arbitrarily. Instead, it is the result of a non-parametric post-hoc calibration method applied on 5 million instances from the CIFAR-5m data set [32]. On a calibration data set so vast, a non-parametric method can learn a very detailed calibration function close to the optimum: the learned result is a close imitation of a realistic true calibration map. The exact procedure used to define $\boldsymbol{c}$ was as follows:

1. ResNet-110 convolutional neural network [11] was trained on 45000 data points from the CIFAR-5m data set [32]. The model was used to predict 5 million images on the data set.
2. The 5 million predicted logit vectors $\boldsymbol{z}$ were scaled down to $\boldsymbol{z}/2.3$ to reduce the complexity of the final result. The temperature 2.3 was chosen arbitrarily: the optimal temperature of temperature scaling would have been 3.
3. The 5 million logit vectors were turned into probabilities with the softmax function. The probabilities were grouped into three class groups to reduce the 10-class problem to a 3-class problem. The class groups were $\{$airplanes, cars, birds, cats$\}$, $\{$deer, dogs, frogs$\}$, $\{$horses, ships, trucks$\}$.
4. The random calibration forest method with 512 bins and 250 trees described in Section 3.2 was applied on the 5 million scaled and grouped data points. The learned result of the random calibration forest is considered the true calibration function $\boldsymbol{c}$ depicted in Figure 14.

The ResNet-110 predictions on the CIFAR-5m data set were provided by Kängsepp et al. [14]; no model training was done in this thesis. CIFAR-5m is not actually a real-world data set: it is generated by sampling a DDPM generative model [12] trained on CIFAR-10 [15], resulting in realistic synthetic instances. This is also why it is so large and contains over 5 million instances. The description of the CIFAR-10 data set, which is imitated by CIFAR-5m, is given in the upcoming real experiments section.

4.2.2 Compared post-hoc calibration methods

In the synthetic experiments, the following post-hoc calibration methods are compared: histogram binning applied with 10 equal-width bins; isotonic regression; beta calibration; temperature scaling (TS); vector scaling; Dirichlet calibration with no regularisation (DIR); the full family of intra order-preserving functions (IOP); decision calibration trained for 10 iterations to achieve calibration with regards to all loss functions with 2 decisions (DEC); random calibration forests with 100 trees and 32 bins (forest); KNN calibration with Euclidean distance and 512 neighbors (KNN); kernel calibration with the Dirichlet kernel and parameter 0.1 (kernel). In addition, the uncalibrated predictions (uncal) and the perfectly calibrated predictions (cal) found with the true calibration function are also included to have a baseline to compare against. The learned calibration maps produced by all the methods have been illustrated in different parts of the thesis in Figures 5, 7, 9, 10, 12.

Note that no logits are available for this synthetically generated calibration task and therefore, all calibration methods meant to be applied on logits are instead applied on $\ln(\hat{\boldsymbol{p}})$. While the softmax function of $\ln(\hat{\boldsymbol{p}})$ still gives back the original predictions $\boldsymbol{\sigma}(\ln(\hat{\boldsymbol{p}})) = \hat{\boldsymbol{p}}$, natural logits available from neural networks carry more information than $\ln(\hat{\boldsymbol{p}})$. Therefore, the synthetic data set approach can be slightly unfair to the logit-based methods. In addition, because of this, matrix scaling and Dirichlet calibration are exactly the same on this synthetic task and only Dirichlet calibration is included of the two. Cross-validation was not used to tune the hyperparameters for the synthetic task: Dirichlet calibration was run without regularisation as the method has few parameters with only 3 classes and overfitting is not a problem; IOP was trained with 40 neurons in both hidden layers.

Table 1. Results of the synthetic experiment. Smaller values are better according to every metric except accuracy.

	uncal	TS	DIR	IOP	DEC	forest	KNN	kernel	cal
confidence CE	0.079_8	0.031_7	0.020_2	0.024_4	0.023_3	0.027_6	0.025_5	$\mathbf{0.017_1}$	0.000
classwise CE	0.056_8	0.028_7	0.015_2	0.021_3	0.023_6	0.022_4	0.022_4	$\mathbf{0.014_1}$	0.000
Brier score	0.492_8	0.481_7	0.479_2	0.480_4	0.479_2	0.480_4	0.480_4	$\mathbf{0.478_1}$	0.477
log-loss	0.828_8	0.799_6	0.794_2	0.795_3	0.799_6	0.795_3	0.797_5	$\mathbf{0.792_1}$	0.790
accuracy	0.624_6	0.624_6	$\mathbf{0.627_1}$	0.624_6	0.626_2	0.625_3	0.625_3	0.625_3	0.627
average loss gap	0.038_8	0.015_7	0.011_3	$\mathbf{0.008_1}$	0.011_3	0.012_5	0.012_5	0.009_2	0.000
average rank	7.7	6.7	2.0	3.5	3.7	4.2	4.3	1.5	

4.2.3 Results

Table 1 compares the methods according to the evaluation metrics listed in Section 4.1. The scores in every row are ranked from best to worst, with the rank displayed as a subindex; the best score in every row is marked in bold. Some of the worse-performing methods have been left out of the table to improve readability.

It can be seen that all calibration methods improve all the metrics over uncalibrated predictions. Accuracy is a small exception: in some cases, it is left unchanged. The metrics are in a strong correlation with each other: the rankings according to different metrics are very similar for most methods. Overall, the best performing method is kernel calibration with Dirichlet calibration following closely as second best. KNN calibration and random forest calibration give average results. Note the phenomenon that Brier score and log-loss are not 0 even for the perfectly calibrated predictions in the last column due to irreducible loss components of proper scoring rules.

4.3 Real data experiments

The goal of the real data experiments is to compare the proposed methods in a more practical setting and see if they can improve state-of-the-art. The following subsections first describe the data sets and models used in the experiments, then discuss the details of methods under comparison, and finally compare the results.

4.3.1 Data sets and models

The experiments on real data use the data sets CIFAR-10 and CIFAR-100 [15]. Both of them contain 60000 small images of size 32x32 pixels and are divided into a model training set of size 45000, a calibration set of size 5000, and a test set of size 10000. CIFAR-10 has 6000 images for each of its 10 classes: airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks. The test set contains exactly 1000 images for each class. The CIFAR-100 data set is similar to CIFAR-10 but it has 100 classes and only 600 images per class. The following are some examples of the CIFAR-100 classes:

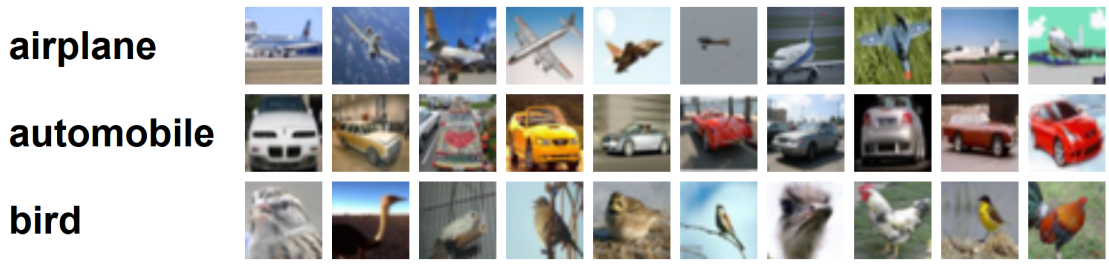

Figure 15. Example images from the CIFAR-10 data set [16].

bed, bridge, cloud, crab, baby, crocodile, hamster, maple, bicycle, and lawn-mower. In Figure 15, there are 10 example images for three of the classes from the CIFAR-10 data set.

Table 2. Data set and model details for real experiments.

			Data set size		
Data set	Models	Classes	Training	Calibration	Test
CIFAR-10	DenseNet-40 ResNet-110 ResNet Wide 32	10	45000	5000	10000
CIFAR-100	DenseNet-40 ResNet-110 ResNet Wide 32	100	45000	5000	10000

On both data sets, the predictions of convolutional neural networks ResNet-110 [11], ResNet Wide 32 [44], DenseNet-40 [13] are used. The logits of these three CNN-s on the calibration and test set were provided by authors of Kull et al. [19]: no actual model training was done for this thesis. An overview of the used data sets and classifiers is given in Table 2.

4.3.2 Compared post-hoc calibration methods

The proposed methods are compared against uncalibrated predictions (no calibration method applied) to have a baseline, Dirichlet calibration with ODIR , matrix scaling with ODIR, vector scaling, temperature scaling, the diagonal subfamily of intra order-preserving functions, decision calibration applied in composition with temperature scaling. As one-vs-rest type methods have been shown to be no longer the state-of-the-art [8], they are not included in the experiments to keep the tables neater.

Of the proposed methods, only KNN calibration with Kullback-Leibler divergence is included in the tables of Section 4.3.3 with the exception of Table 4 which also includes kernel calibration with Dirichlet kernel and random calibration forests. The tables in Appendix I also include comparisons between all the proposed methods: KNN calibration with Euclidean distance and Kullback-Leibler divergence; kernel calibration with RBF and Dirichlet kernel; and random forest calibration. Only KNN calibration with Kullback-Leibler divergence is included in the tables of Section 4.3.3 as it is the best performing one among the proposed methods. An overview of the compared methods for real data experiments is listed in Table 3. The table also describes the hyperparameter details for the methods.

Table 3. Details of compared post-hoc calibration methods for real experiments.

Method	**Short name**	**Details**
uncalibrated	uncal	baseline results of classifier with no calibration method applied
Dirichlet calibration	DIR	both trained with ODIR; method implementation and best hyperparameters for the data set classifier combinations were provided by Kull et al. [19]
matrix scaling	MS	
vector scaling	VS	-
intra order-preserving functions	IOP	diagonal subfamily of intra-order preserving functions; method implementation and best hyperparameters for the data set classifier combinations were obtained from Rahimi et al. [38]
temperature scaling	TS	-
decision calibration	DEC	applied in composition with temperature scaling; trained to achieve decision calibration with respect to all loss functions with 2 decisions; number of trained iterations determined unfairly by looking at test set Brier score to save resources avoiding cross-validation; the final output was normalised to sum to 1 which was not done in the original implementation
random calibration forest	forest	method fit with 250 trees; number of bins found from $[2^1, 2^2, \ldots, 2^{12}]$ with 10-fold cross-validation with log-loss
k-nearest-neighbors calibration	KNN	number of neighbors found from $[2^5, 2^6, \ldots, 2^{12}]$ with 10-fold cross-validation with log-loss; KNN$_{euc}$ uses Euclidean distance, KNN$_{kl}$ or simply KNN uses Kullback-Leibler divergence
kernel calibration	kernel	kernel parameter found from $[1.6 \cdot 2^{-9}, 1.6 \cdot 2^{-8}, \ldots, 1.6 \cdot 2^{6}]$ with 10-fold cross-validation with log-loss; kernel$_{RBF}$ is with RBF kernel, kernel$_{DIR}$ or simply kernel is with Dirichlet kernel

4.3.3 Results

Table 4 compares the proposed non-parametric methods on DenseNet-40 CIFAR-10 against the simplest parametric method temperature scaling. Similar to synthetic experiments, the scores in every row are ranked from best to worst, with the rank displayed as a subindex; the best scores in every row are marked in bold. It can be seen that the proposed methods all improve Brier score, log-loss, confidence ECE, and average loss gap; and maintain accuracy, classwise ECE. There is one exception: kernel calibration slightly worsens classwise ECE. Compared with the synthetic experiments where the proposed non-parametric methods all performed well and kernel calibration was even the best performing method, they are now all considerably worse: they all lose by far to the simple parametric method temperature scaling. This is understandable as while

Table 4. Temperature scaling versus the proposed methods on DenseNet-40 CIFAR-10. Smaller values are better according to every metric except accuracy.

	uncal	TS	forest	KNN	kernel
confidence ECE	5.493_5	$\mathbf{0.924_1}$	3.201_2	3.342_3	3.381_4
classwise ECE	0.445_4	$\mathbf{0.255_1}$	0.436_3	0.393_2	0.461_5
Brier score	0.127_5	$\mathbf{0.110_1}$	0.120_3	0.118_2	0.124_4
log-loss	0.428_5	$\mathbf{0.225_1}$	0.292_2	0.298_4	0.293_3
accuracy	$\mathbf{0.924_1}$	$\mathbf{0.924_1}$	$\mathbf{0.924_1}$	$\mathbf{0.924_1}$	$\mathbf{0.924_1}$
loss gap	0.030_5	$\mathbf{0.006_1}$	0.011_2	0.016_4	0.015_3
average rank	4.2	1	2.2	2.7	3.3

the calibration data set size in both cases is 5000, the number of classes in the synthetic experiment was 3, and now the number of classes is 10. It can be expected that the proposed methods perform even worse on CIFAR-100 where the number of classes is 100. Good non-parametric estimates of calibration error with more classes require considerably more data points. In addition, with more classes, the probability space is more sparsely populated and the assumption of equal calibration errors for neighbors becomes less sensible. As in a direct comparison the proposed methods are clearly outperformed, then in the following tables on real data, the methods are applied in composition with temperature scaling to see if the composition can improve the state-of-the-art. The composition of post-hoc calibration methods was discussed in Section 3.5.

Expected calibration errors. Table 5 presents the results for confidence ECE. Note that the notation in the last two columns denotes that decision and KNN calibration have been applied in composition with temperature scaling (temperature scaling is applied first). On CIFAR-10 the best method according to confidence ECE is clearly KNN calibration having an average rank of 1 and offering considerable improvements over the competitors. For example, on DenseNet-40 the KNN method achieves confidence ECE 0.52, while the second-best method IOP gives ECE score of 0.80. On CIFAR-100 the best method according to confidence ECE is temperature scaling.

Table 6 displays the results for classwise ECE. The best results are obtained by Dirichlet calibration, matrix scaling, and vector scaling which all have very similar scores. For example, on CIFAR-100 ResNet Wide 32, all the three methods share a score of 0.09. KNN calibration comes close to the best-performing methods on CIFAR-10, improving the results of temperature scaling. ResNet-110 CIFAR-10 is an outlier where KNN calibration is also the best for classwise ECE achieving the score 0.17. On CIFAR-100 the KNN method worsens the result of temperature scaling. Note that classwise ECE values tend to be a lot smaller than confidence ECE: this is due to the fact that most predictions coming from the softmax function are tiny and wash out the ECE score [34].

Table 5. Confidence ECE ($\times 10^2$).

								TS	
		uncal	DIR	MS	VS	IOP	TS	DEC	KNN
C-10	DenseNet-40	5.49_8	0.94_5	0.91_3	1.00_6	0.80_2	0.92_4	1.20_7	$\mathbf{0.52_1}$
	ResNet-110	4.75_8	1.10_7	0.99_5	1.07_6	0.91_2	0.94_3	0.94_3	$\mathbf{0.84_1}$
	ResNet Wide 32	4.48_8	0.70_4	0.75_5	0.78_6	0.69_2	0.69_2	0.89_7	$\mathbf{0.54_1}$
	average rank	8	5.3	4.3	6	2	3	5.7	1
C-100	DenseNet-40	21.16_8	0.95_2	1.22_4	0.96_3	3.45_6	$\mathbf{0.79_1}$	3.67_7	3.34_5
	ResNet-110	18.48_8	2.27_3	2.31_4	3.04_6	2.79_5	2.13_2	3.18_7	$\mathbf{1.90_1}$
	ResNet Wide 32	18.78_8	1.54_3	1.85_5	1.68_4	$\mathbf{1.03_1}$	1.41_2	3.08_7	2.46_6
	average rank	8	2.7	4.3	4.3	4	1.7	7	4

Table 6. Classwise ECE ($\times 10^2$).

								TS	
		uncal	DIR	MS	VS	IOP	TS	DEC	KNN
C-10	DenseNet-40	0.44_8	$\mathbf{0.21_1}$	$\mathbf{0.21_1}$	0.23_3	0.25_5	0.25_5	0.30_7	0.24_4
	ResNet-110	0.36_8	0.20_3	0.18_2	0.20_3	0.21_5	0.22_6	0.23_7	$\mathbf{0.17_1}$
	ResNet Wide 32	0.50_8	0.19_3	$\mathbf{0.18_1}$	$\mathbf{0.18_1}$	0.45_6	0.45_6	0.29_5	0.27_4
	average rank	8	2.3	1.3	2.3	5.3	5.7	6.3	3
C-100	DenseNet-40	0.17_8	$\mathbf{0.09_1}$	0.10_3	$\mathbf{0.09_1}$	0.11_5	0.10_3	0.14_6	0.15_7
	ResNet-110	0.13_6	0.10_2	0.11_4	0.11_4	0.10_2	$\mathbf{0.09_1}$	0.13_6	0.13_6
	ResNet Wide 32	0.12_6	$\mathbf{0.09_1}$	$\mathbf{0.09_1}$	$\mathbf{0.09_1}$	0.10_4	0.10_4	0.13_7	0.14_8
	average rank	6.7	1.3	2.7	2	3.7	2.7	6.3	7

Proper scoring rules. Table 7 displays the results for Brier score and Table 8 the results for log-loss. The best method according to both proper scoring rules is matrix scaling. For Brier score, KNN calibration offers minor improvements over temperature scaling on the smaller CIFAR-10 data set but maintains the score on CIFAR-100. Overall, for Brier score the differences on CIFAR-10 are small. For example, on CIFAR-10 DenseNet-40, all the methods get the same score of 0.110. According to log-loss, there is no benefit of applying KNN calibration in composition with temperature scaling: for CIFAR-10 log-loss lowers slightly, and for CIFAR-100 log-loss lowers considerably. Decision calibration which is also applied in composition with temperature scaling manages to improve Brier score but suffers from a large decrease in log-loss.

Table 7. Brier score.

								TS	
		uncal	DIR	MS	VS	IOP	TS	DEC	KNN
C-10	DenseNet-40	0.127_{8}	$\mathbf{0.110_{1}}$	$\mathbf{0.110_{1}}$	$\mathbf{0.110_{1}}$	$\mathbf{0.110_{1}}$	$\mathbf{0.110_{1}}$	$\mathbf{0.110_{1}}$	$\mathbf{0.110_{1}}$
	ResNet-110	0.110_{8}	0.098_{4}	0.098_{4}	0.098_{4}	$\mathbf{0.097_{1}}$	0.098_{4}	$\mathbf{0.097_{1}}$	$\mathbf{0.097_{1}}$
	ResNet Wide 32	0.105_{8}	$\mathbf{0.089_{1}}$	$\mathbf{0.089_{1}}$	$\mathbf{0.089_{1}}$	0.092_{6}	0.092_{6}	$\mathbf{0.089_{1}}$	0.090_{5}
	average rank	8	2	2	2	2.7	3.7	1	2.3
C-100	DenseNet-40	0.491_{8}	0.399_{2}	$\mathbf{0.398_{1}}$	0.403_{6}	0.403_{6}	0.401_{3}	0.401_{3}	0.402_{5}
	ResNet-110	0.453_{8}	0.391_{2}	0.391_{2}	0.394_{7}	0.391_{2}	0.392_{5}	$\mathbf{0.39_{1}}$	0.392_{5}
	ResNet Wide 32	0.432_{8}	0.353_{3}	$\mathbf{0.351_{1}}$	0.352_{2}	0.355_{5}	0.355_{5}	0.354_{4}	0.355_{5}
	average rank	8	2.3	1.3	5	4.3	4.3	2.7	5

Table 8. Log-loss.

								TS	
		uncal	DIR	MS	VS	IOP	TS	DEC	KNN
C-10	DenseNet-40	0.428_{8}	0.223_{2}	$\mathbf{0.222_{1}}$	0.223_{2}	0.225_{4}	0.225_{4}	0.264_{7}	0.229_{6}
	ResNet-110	0.358_{8}	0.205_{2}	$\mathbf{0.204_{1}}$	0.206_{4}	0.208_{5}	0.209_{6}	0.232_{7}	0.205_{2}
	ResNet Wide 32	0.382_{8}	0.183_{2}	$\mathbf{0.182_{1}}$	0.183_{2}	0.192_{6}	0.191_{5}	0.244_{7}	0.190_{4}
	average rank	8	2	1	2.7	5	5	7	4
C-100	DenseNet-40	2.017_{8}	1.057_{2}	$\mathbf{1.047_{1}}$	1.062_{4}	1.067_{5}	1.057_{2}	1.422_{7}	1.116_{6}
	ResNet-110	1.694_{8}	1.094_{4}	$\mathbf{1.073_{1}}$	1.093_{3}	1.106_{5}	1.092_{2}	1.466_{7}	1.150_{6}
	ResNet Wide 32	1.802_{8}	0.950_{5}	$\mathbf{0.931_{1}}$	0.941_{2}	0.945_{3}	0.945_{3}	1.276_{7}	0.994_{6}
	average rank	8	3.7	1	3	4.3	2.3	7	6

Accuracy. Table 7 includes the results for classification accuracy. The fluctuations in accuracy are very small, in a few cases some methods reduce accuracy by 0.001, most of the time accuracy is preserved or increased by a tiny amount. Decision calibration seems to perform better than other methods even though the differences are minor.

Average loss gap. Table 7 includes the results for average loss gap. On CIFAR-10 the best methods are decision and KNN calibration with corresponding average ranks of 1.7 and 2.3. On ResNet Wide 32 CIFAR-10, Dirichlet calibration, matrix scaling, and vector scaling also all perform well. On CIFAR-100 both decision and KNN calibration worsen the result of temperature scaling. The best method for CIFAR-100 is temperature scaling. IOP could also be considered the best for CIFAR-100 if the failed result of 0.0197 on DenseNet-40 is considered an exception.

Table 9. Accuracy.

								TS	
		uncal	DIR	MS	VS	IOP	TS	DEC	KNN
C-10	DenseNet-40	0.924_{6}	$\mathbf{0.925_{1}}$	$\mathbf{0.925_{1}}$	$\mathbf{0.925_{1}}$	0.924_{6}	0.924_{6}	$\mathbf{0.925_{1}}$	$\mathbf{0.925_{1}}$
	ResNet-110	$\mathbf{0.936_{1}}$	0.935_{7}	$\mathbf{0.936_{1}}$	0.935_{7}	$\mathbf{0.936_{1}}$	$\mathbf{0.936_{1}}$	$\mathbf{0.936_{1}}$	$\mathbf{0.936_{1}}$
	ResNet Wide 32	0.939_{6}	$\mathbf{0.942_{1}}$	$\mathbf{0.942_{1}}$	$\mathbf{0.942_{1}}$	0.939_{6}	0.939_{6}	$\mathbf{0.942_{1}}$	0.940_{5}
	average rank	4.3	3	1	3	4.3	4.3	1	2.3
C-100	DenseNet-40	0.700_{4}	0.703_{2}	$\mathbf{0.704_{1}}$	0.699_{8}	0.700_{4}	0.700_{4}	0.703_{2}	0.700_{4}
	ResNet-110	0.715_{4}	$\mathbf{0.717_{1}}$	0.715_{4}	0.716_{2}	0.715_{4}	0.715_{4}	0.716_{2}	0.715_{4}
	ResNet Wide 32	0.738_{5}	0.740_{2}	0.740_{2}	0.738_{5}	0.738_{5}	0.738_{5}	$\mathbf{0.741_{1}}$	0.739_{4}
	average rank	4.3	1.7	2.3	5	4.3	4.3	1.7	4

Table 10. Average loss gap.

								TS	
		uncal	DIR	MS	VS	IOP	TS	DEC	KNN
C-10	DenseNet-40	0.0295_{8}	0.0055_{3}	0.0058_{4}	0.0061_{6}	0.0060_{5}	0.0063_{7}	0.0051_{2}	$\mathbf{0.0046_{1}}$
	ResNet-110	0.0259_{8}	0.0060_{6}	0.0058_{5}	0.0062_{7}	0.0045_{3}	0.0056_{4}	0.0042_{2}	$\mathbf{0.0035_{1}}$
	ResNet Wide 32	0.0251_{8}	0.0048_{3}	0.0046_{2}	0.0048_{3}	0.0098_{7}	0.0095_{6}	$\mathbf{0.0045_{1}}$	0.0049_{5}
	average rank	8	4	3.7	5.3	5	5.7	1.7	2.3
C-100	DenseNet-40	0.1154_{8}	$\mathbf{0.0064_{1}}$	0.0077_{4}	0.0068_{2}	0.0197_{6}	0.0071_{3}	0.0177_{5}	0.0210_{7}
	ResNet-110	0.1008_{8}	0.0096_{3}	0.0109_{5}	0.0119_{7}	$\mathbf{0.0059_{1}}$	0.0078_{2}	0.0117_{6}	0.0098_{4}
	ResNet Wide 32	0.1027_{8}	0.0086_{5}	0.0085_{3}	0.0085_{3}	$\mathbf{0.0068_{1}}$	0.0077_{2}	0.0115_{6}	0.0141_{7}
	average rank	8	3	4	4	2.7	2.3	5.7	6

Conclusion. For every model trained on CIFAR-10, KNN calibration offers improvements over temperature scaling according to every metric with a small exception for log-loss. Measured by confidence ECE or average loss gap, KNN calibration applied in composition with temperature scaling is the best method or is tied with the best. On CIFAR-100, there is no benefit of applying KNN calibration in composition with temperature scaling: for most metrics, it makes the result considerably worse.

The proposed non-parametric methods fail on CIFAR-100 as it has more classes and therefore two problems arise: first, the larger the output vector, the more neighbors are needed to get a decent estimate for the calibration error by averaging the difference between labels and predictions of neighbors; second, in high dimensions, the probability space is more sparsely populated and data points start to have roughly the same distance from each other [1].

5 Conclusion

This thesis explored the field of post-hoc calibration methods for multi-class classifiers. Several non-parametric methods were proposed: random forest calibration, KNN calibration, and kernel calibration. All the proposed methods assume that the calibration error of a data point can be modeled by the calibration errors of its neighbors. The assumption results in using the average difference of predictions and labels in a close neighborhood to estimate the calibration error of a prediction. Based on the definition of calibration, the found calibration error estimate is then subtracted from the prediction to reach a calibrated prediction.

An illustrative synthetic experiment was carried out which showed the potential of the proposed methods and compared the calibration maps learned by different algorithms. In addition, experiments were carried out on three convolutional neural networks trained on CIFAR-10 and CIFAR-100 to compare the proposed methods with their competitors in a real-world setting. The experimental results on real data showed that the proposed methods alone are clearly not competitive for data sets with many classes and small calibration sets. The composition of the proposed methods with temperature scaling was also compared. Results on CIFAR-10 suggest that there is merit in using the proposed methods in composition with temperature scaling. The best proposed method, KNN calibration applied in composition with temperature scaling, topped the state-of-the-art in confidence calibration and was comparable in many other metrics. Results on CIFAR-100, however, clearly show the limitations of the proposed non-parametric approach. With a few marginal exceptions, the methods fail completely on the data set due to the limited amount of calibration data.

For future work, the limits of the proposed methods could be studied more thoroughly. How does the border between improvement and deterioration in calibration depend on the number of classes in the data set; the calibration data set size; or the distribution of the predictions on the probability simplex? In addition, the composition of different calibration methods could be studied further as this work and several previous [45, 46] have shown the possible benefits.

References

[1] Charu C. Aggarwal, Alexander Hinneburg, and Daniel A. Keim. On the Surprising Behavior of Distance Metrics in High Dimensional Space. In *Database Theory — ICDT 2001*, pages 420–434, 2001.

[2] Arsenii Ashukha, Alexander Lyzhov, Dmitry Molchanov, and Dmitry P. Vetrov. Pitfalls of In-Domain Uncertainty Estimation and Ensembling in Deep Learning. In *International Conference on Learning Representations*, 2020.

[3] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006.

[4] Glenn W. Brier. Verification of Forecasts Expressed in Terms of Probability. *Monthly Weather Review*, 78(1):1–3, 1950.

[5] 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.

[6] Morris H. DeGroot and Stephen E. Fienberg. The Comparison and Evaluation of Forecasters. *Journal of the Royal Statistical Society. Series D (The Statistician)*, 32 (1/2):12–22, 1983.

[7] Pedro M. Domingos and Michael J. Pazzani. Beyond Independence: Conditions for the Optimality of the Simple Bayesian Classifier. In *ICML*, 1996.

[8] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On Calibration of Modern Neural Networks. In *International Conference on Machine Learning*, pages 1321–1330, 2017.

[9] Chirag Gupta and Aaditya Ramdas. Top-label calibration and multiclass-to-binary reductions. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=WqoBaaPHS-. Last visited 2022-05-12.

[10] Kartik Gupta, Amir Rahimi, Thalaiyasingam Ajanthan, Thomas Mensink, Cristian Sminchisescu, and Richard Hartley. Calibration of Neural Networks using Splines. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=eQe8DEWNN2W. Last visited 2022-05-12.

[11] 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*, pages 770–778, 2016.

[12] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising Diffusion Probabilistic Models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

[13] 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*, pages 4700–4708, 2017.

[14] Markus Kängsepp, Kaspar Valk, and Meelis Kull. On the Usefulness of the Fit-on-the-Test View on Evaluating Calibration of Classifiers. *arXiv preprint arXiv:2203.08958*, 2022.

[15] Alex Krizhevsky. Learning Multiple Layers of Features from Tiny Images. 2009.

[16] Alex Krizhevsky. The CIFAR-10 dataset. https://www.cs.toronto.edu/~kriz/cifar.html, 2009. Last visited 2022-05-12.

[17] Meelis Kull and Peter Flach. Novel Decompositions of Proper Scoring Rules for Classification: Score Adjustment as Precursor to Calibration. In *Machine Learning and Knowledge Discovery in Databases*, pages 68–85, 2015.

[18] Meelis Kull, Telmo Silva Filho, and Peter Flach. Beta calibration: a well-founded and easily implemented improvement on logistic calibration for binary classifiers. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, pages 623–631, 2017.

[19] 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.

[20] Ananya Kumar, Percy S Liang, and Tengyu Ma. Verified Uncertainty Calibration. *Advances in Neural Information Processing Systems*, 32, 2019.

[21] Aviral Kumar, Sunita Sarawagi, and Ujjwal Jain. Trainable Calibration Measures for Neural Networks from Kernel Mean Embeddings. In *International Conference on Machine Learning*, pages 2805–2814, 2018.

[22] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles. *Advances in neural information processing systems*, 30, 2017.

[23] Christian Leibig, Vaneeda Allken, Murat Seçkin Ayhan, Philipp Berens, and Siegfried Wahl. Leveraging uncertainty information from deep neural networks for disease detection. *Scientific reports*, 7(1):1–14, 2017.

[24] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal Loss for Dense Object Detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[25] Wesley J Maddox, Pavel Izmailov, Timur Garipov, Dmitry P Vetrov, and Andrew Gordon Wilson. A Simple Baseline for Bayesian Uncertainty in Deep Learning. *Advances in Neural Information Processing Systems*, 32, 2019.

[26] Edgar C Merkle and Mark Steyvers. Choosing a Strictly Proper Scoring Rule. *Decision Analysis*, 10(4):292–304, 2013.

[27] Jishnu Mukhoti, Viveka Kulharia, Amartya Sanyal, Stuart Golodetz, Philip Torr, and Puneet Dokania. Calibrating Deep Neural Networks using Focal Loss. *Advances in Neural Information Processing Systems*, 33:15288–15299, 2020.

[28] Rafael Müller, Simon Kornblith, and Geoffrey E Hinton. When Does Label Smoothing Help? *Advances in neural information processing systems*, 32, 2019.

[29] Allan H. Murphy and Robert L. Winkler. Reliability of Subjective Probability Forecasts of Precipitation and Temperature. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 26(1):41–47, 1977.

[30] Kevin P. Murphy. *Probabilistic Machine Learning: An introduction*. MIT Press, 2022. URL probml.ai. Last visited 2022-05-12.

[31] Mahdi Pakdaman Naeini, Gregory Cooper, and Milos Hauskrecht. Obtaining Well Calibrated Probabilities Using Bayesian Binning. In *AAAI Conference on Artificial Intelligence*, 2015.

[32] Preetum Nakkiran, Behnam Neyshabur, and Hanie Sedghi. The Deep Bootstrap Framework: Good Online Learners are Good Offline Generalizers. In *International Conference on Learning Representations*, 2021.

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

[34] Jeremy Nixon, Michael W Dusenberry, Linchuan Zhang, Ghassen Jerfel, and Dustin Tran. Measuring Calibration in Deep Learning. In *CVPR Workshops*, volume 2, 2019.

[35] Frédéric Ouimet and Raimon Tolosana-Delgado. Asymptotic properties of Dirichlet kernel density estimators. *Journal of Multivariate Analysis*, 187:104832, 2022.

[36] John C. Platt. Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods. In *Advances in large margin classifiers*, pages 61–74, 1999.

[37] Teodora Popordanoska, Raphael Sayer, and Matthew B Blaschko. Calibration Regularized Training of Deep Neural Networks using Kernel Density Estimation. 2021. URL https://openreview.net/forum?id=1-lFH8oYTI. Last visited 2022-05-12.

[38] Amir Rahimi, Amirreza Shaban, Ching-An Cheng, Richard Hartley, and Byron Boots. Intra Order-Preserving Functions for Calibration of Multi-Class Neural Networks. *Advances in Neural Information Processing Systems*, 33:13456–13467, 2020.

[39] Rebecca Roelofs, Nicholas Cain, Jonathon Shlens, and Michael C Mozer. Mitigating Bias in Calibration Error Estimation. In *International Conference on Artificial Intelligence and Statistics*, pages 4036–4054, 2022.

[40] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the Inception Architecture for Computer Vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[41] 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*, pages 3459–3467, 2019.

[42] Bianca Zadrozny and Charles Elkan. Obtaining Calibrated Probability Estimates from Decision Trees and Naive Bayesian Classifiers. *ICML*, 1, 2001.

[43] 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*, pages 694–699, 2002.

[44] Sergey Zagoruyko and Nikos Komodakis. Wide Residual Networks. In *Proceedings of the British Machine Vision Conference (BMVC)*, pages 87.1–87.12, 2016.

[45] Jize Zhang, Bhavya Kailkhura, and T Yong-Jin Han. Mix-n-Match: Ensemble and Compositional Methods for Uncertainty Calibration in Deep Learning. In *International Conference on Machine Learning*, pages 11117–11128, 2020.

[46] Shengjia Zhao, Michael P Kim, Roshni Sahoo, Tengyu Ma, and Stefano Ermon. Calibrating Predictions to Decisions: A Novel Approach to Multi-Class Calibration. In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021.

Appendix

I. Additional comparisons on real data

Table 11. Confidence ECE ($\times 10^2$). Comparison between proposed methods.

				TS				
		uncal	TS	KNN$_{kl}$	KNN$_{euc}$	forest	kernel$_{RBF}$	kernel$_{DIR}$
C-10	DenseNet-40	5.49_7	0.92_5	$\mathbf{0.52_1}$	0.72_2	0.92_5	0.80_3	0.82_4
	ResNet-110	4.75_7	0.94_5	$\mathbf{0.84_1}$	0.88_2	0.90_3	1.14_6	0.92_4
	ResNet Wide 32	4.48_7	0.69_2	$\mathbf{0.54_1}$	0.71_3	0.91_5	0.97_6	0.86_4
	average rank	7	4	1	2.3	4.3	5	4

Table 12. Classwise ECE ($\times 10^2$). Comparison between proposed methods.

				TS				
		uncal	TS	KNN$_{kl}$	KNN$_{euc}$	forest	kernel$_{RBF}$	kernel$_{DIR}$
C-10	DenseNet-40	0.44_7	0.25_4	0.24_2	0.24_2	0.25_4	0.26_6	$\mathbf{0.23_1}$
	ResNet-110	0.36_7	0.22_5	$\mathbf{0.17_1}$	0.18_2	0.19_4	0.22_5	0.18_2
	ResNet Wide 32	0.50_7	0.45_6	$\mathbf{0.27_1}$	0.30_3	0.29_2	0.37_4	0.37_4
	average rank	7	5	1.3	2.3	3.3	5	2.3

Table 13. Brier score. Comparison between proposed methods.

				TS				
		uncal	TS	KNN$_{kl}$	KNN$_{euc}$	forest	kernel$_{RBF}$	kernel$_{DIR}$
C-10	DenseNet-40	0.127_7	$\mathbf{0.110_1}$	$\mathbf{0.110_1}$	$\mathbf{0.110_1}$	$\mathbf{0.110_1}$	$\mathbf{0.110_1}$	$\mathbf{0.110_1}$
	ResNet-110	0.11_7	0.098_6	$\mathbf{0.097_1}$	$\mathbf{0.097_1}$	$\mathbf{0.097_1}$	$\mathbf{0.097_1}$	$\mathbf{0.097_1}$
	ResNet Wide 32	0.105_7	0.092_6	0.090_2	0.090_2	$\mathbf{0.089_1}$	0.090_2	0.091_5
	average rank	7	4.3	1.3	1.3	1	1.3	2.3

Table 14. Log-loss. Comparison between proposed methods.

				TS				
		uncal	TS	KNN$_{kl}$	KNN$_{euc}$	forest	kernel$_{RBF}$	kernel$_{DIR}$
C-10	DenseNet-40	0.428_7	$\mathbf{0.225_1}$	0.229_5	0.228_4	0.227_3	0.229_5	$\mathbf{0.225_1}$
	ResNet-110	0.358_7	0.209_6	$\mathbf{0.205_1}$	0.206_2	0.207_4	0.208_5	0.206_2
	ResNet Wide 32	0.382_7	0.191_5	0.190_4	$\mathbf{0.189_1}$	$\mathbf{0.189_1}$	0.192_6	$\mathbf{0.189_1}$
	average rank	7	4	3.3	2.3	2.7	5.3	1.3

Table 15. Accuracy. Comparison between proposed methods.

				TS				
		uncal	TS	KNN$_{kl}$	KNN$_{euc}$	forest	kernel$_{RBF}$	kernel$_{DIR}$
C-10	DenseNet-40	0.924_5	0.924_5	0.925_2	0.925_2	0.924_5	0.925_2	$\mathbf{0.926_1}$
	ResNet-110	$\mathbf{0.936_1}$	$\mathbf{0.936_1}$	$\mathbf{0.936_1}$	$\mathbf{0.936_1}$	$\mathbf{0.936_1}$	$\mathbf{0.936_1}$	$\mathbf{0.936_1}$
	ResNet Wide 32	0.939_6	0.939_6	0.940_4	0.940_4	$\mathbf{0.942_1}$	$\mathbf{0.942_1}$	$\mathbf{0.942_1}$
	average rank	4	4	2.3	2.3	2.3	1.3	1

Table 16. Average loss gap. Comparison between proposed methods.

				TS				
		uncal	TS	KNN$_{kl}$	KNN$_{euc}$	forest	kernel$_{RBF}$	kernel$_{DIR}$
C-10	DenseNet-40	0.0295_7	0.0063_6	0.0046_3	0.0046_3	$\mathbf{0.0040_1}$	$\mathbf{0.0040_1}$	0.0054_5
	ResNet-110	0.0259_7	0.0056_6	0.0035_4	0.0033_2	0.0035_4	$\mathbf{0.0032_1}$	0.0034_3
	ResNet Wide 32	0.0251_7	0.0095_6	0.0049_2	0.0056_3	$\mathbf{0.0047_1}$	0.0058_4	0.0075_5
	average rank	7	6	3	2.7	2	2	4.3

II. Licence

Non-exclusive licence to reproduce thesis and make thesis public

I, **Kaspar Valk**,

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to

   reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

   Calibration of Multi-Class Probabilistic Classifiers,

   supervised by Meelis Kull.

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Kaspar Valk
17/05/2022