Simple Non-Parametric Calibration of Multi-Class Classifiers

Anonymous authors
Paper under double-blind review

Abstract

A probabilistic classifier is considered calibrated if it outputs probabilities that are in correspondence with empirical class proportions. Calibration is essential in safety-critical tasks where small deviations between the predicted probabilities and the actually observed class proportions can incur high 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 work explores the field of post-hoc calibration methods for multi-class classifiers and proposes a novel non-parametric post-hoc calibration method. The basis of the proposed method is the assumption of locally equal calibration errors on the probability simplex. This assumption has been previously used but never clearly stated in the calibration literature. The proposed calibration method is shown to offer improvements to the state-of-the-art according to the expected calibration error metric on CIFAR-10 and CIFAR-100 data sets.

1 Introduction

Probabilistic classifiers take as input some data and produce as output probability distributions over classes. For example, a classifier could be tasked to take as input an X-ray image of a person's chest and produce as output a vector of three probabilities for whether the image depicts a *healthy lung*, *lung cancer* or *some other lung disease*. It is possible 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 (Guo et al., 2017; Lakshminarayanan et al., 2017). Distorted output probabilities are also characteristic of many classical machine learning methods such as naive Bayes or decision trees (Niculescu-Mizil & Caruana, 2005; Domingos & Pazzani, 1996).

Well-calibrated classifiers are essential in safety-critical applications (e.g. medical diagnosis, autonomous driving) where small deviations of predicted probabilities from being calibrated can cause costly mistakes (Leibig et al., 2017). 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. (2020) studied the use of focal loss (Lin et al., 2017) instead of log-loss for training better calibrated classifiers; Müller et al. (2019) investigated the use of label smoothing (Szegedy et al., 2016) during training for better calibration; Kumar et al. (2018) and Popordanoska et al. (2021) proposed additional terms to be added to the training time loss function that penalize miscalibration; Maddox et al. (2019) 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 validation

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 (Platt, 1999); isotonic calibration (Zadrozny & Elkan, 2002); histogram binning (Zadrozny & Elkan, 2001); beta calibration (Kull et al., 2017). 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 (Zadrozny & Elkan, 2002). 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. (2017) proposed temperature scaling, vector scaling, and matrix scaling; Kull et al. (2019) introduced Dirichlet calibration; Zhao et al. (2021) proposed a method specifically intended for decision making scenarios; Rahimi et al. (2020) suggested intra-order preserving functions; Wenger et al. (2020) proposed a non-parametric method based on a latent Gaussian process.

This work takes a look at the *post-hoc calibration method* approach for achieving better calibrated multi-class classifiers. While there already exist many strong multi-class methods, several of them are limited to symmetrical transformations for all the classes; for example, temperature scaling, Gaussian process calibration, diagonal and order-invariant subfamilies of the intra-order preserving functions are all symmetrical. As shown in the experiments section, symmetrical transformations are usually not a problem but can be severely limiting in some cases. The asymmetrical existing methods are limited in their expressivity; for example, matrix scaling and vector scaling are limited to only linear transformations in the logit space. This work proposes an intuitive and simple non-parametric post-hoc calibration method that is not limited by a symmetrical transformation or the expressivity of parametric function families. The basis of the proposed method is assuming that similar predictions on the probability simplex have similar calibration errors. The method is shown to outperform competing methods, offering improvements in expected calibration error and avoiding the failures of symmetric methods.

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, the contributions of this work are described and a post-hoc calibration method is proposed. In Section 4, experiments are carried out to compare the proposed method to its competitors.

2 Background and related work

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 work focuses on m-class classification problems with feature space $\mathcal{X}$ and one-hot encoded label space $\mathcal{Y} = \{(1, 0, \ldots, 0), (0, 1, \ldots, 0), \ldots, (0, 0, \ldots, 1)\}$, where $m \geq 3$. 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.

2.2 Calibration

There exist several definitions for calibration in the context of probabilistic multi-class classifiers.

Multi-class calibration A classifier is considered to be multi-class-calibrated (or just calibrated) (Kull et al., 2019) if for any prediction vector $\hat{\boldsymbol{p}} \in \Delta^m$ it holds that

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

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

Classwise calibration A weaker notion of classwise calibration conditions the expectation on each class separately (Zadrozny & Elkan, 2002). 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 (Vaicenavicius et al., 2019).

Confidence calibration Another weaker notion, confidence calibration used by Guo et al. (2017) requires calibration only for the predictions of the class with the highest probability in each output. A classifier is considered to be confidence calibrated if for any real value $c \in [0, 1]$ it holds that

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

2.3 Calibration error

Calibration error describes the difference between the predicted probabilities of the classifier and the corresponding perfectly calibrated class probabilities. Kumar et al. (2019) 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, the error defined in such a way is a vector of real values.

2.4 Calibration evaluation

In any real-world setting, true calibration error can not be directly found, it can only be estimated. Common metrics to evaluate calibration are *expected calibration error* (ECE) and proper scoring rules.

The ECE metric groups similar predictions into bins and uses bin averages to estimate the calibration error. For confidence calibration, ECE is calculated using $Y_{argmax\, \hat{\boldsymbol{P}}}$ and $max\, \hat{\boldsymbol{P}}$ and it is defined as

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

where b is the number of bins, $|B_i|$ the number of data points in bin B_i, n the number of data points, $\overline{p}_i$ the average prediction value in the i-th bin and $\overline{y}_i$ the average label value in the i-th bin (Naeini et al., 2015).

For classwise calibration, ECE is defined as

$$\text{classwise ECE} = \frac{1}{m} \sum_{j=1}^{m} \text{class-}j\text{-ECE},$$

where class-j-ECE is calculated with the same formula as in Equation (1) but with values Y_j and $\hat{P}_j$ used instead of $Y_{argmax\,\hat{\boldsymbol{P}}}$ and $max\,\hat{\boldsymbol{P}}$ when calculating $\overline{y}_i$ and $\overline{p}_i$ (Kull et al., 2019).

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. Good metrics to consider in addition to ECE are proper scoring rules (Brier score or log-loss) as they are shown to decompose into calibration loss and refinement loss (Kull & Flach, 2015).

2.5 Post-hoc calibration methods

Calibration of an already trained classifier can be improved by post-hoc calibration methods. 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 a better-calibrated output $\hat{\boldsymbol{c}}(\hat{\boldsymbol{p}})$. The ideal result would be if the calibration method learns the true calibration map $\boldsymbol{c}(\hat{\boldsymbol{p}}) = \mathbb{E}\left[\boldsymbol{Y}|\hat{\boldsymbol{P}} = \hat{\boldsymbol{p}}\right]$. The transformation is typically learned by optimizing a proper scoring rule (Brier score or log-loss) (Rahimi et al., 2020).

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 (Zadrozny & Elkan, 2002). 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 considerable 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 which deforms the calibrated probabilities. One simple binary calibration method that has been commonly applied in a one-versus-rest approach is histogram binning (Zadrozny & Elkan, 2001): it divides the probability space into bins and in each bin sets the calibrated value of predictions belonging to that bin equal to the empirical class distribution value in the bin.

Temperature scaling Temperature scaling is a logit scaling method designed for neural networks introduced by Guo et al. (2017). The method is defined as $\hat{\boldsymbol{c}}(\boldsymbol{z}) = \boldsymbol{\sigma}(\boldsymbol{z}/t)$ where $\boldsymbol{z}$ is the logit vector and $t \in (0, \infty)$ 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.

Matrix and vector scaling Matrix and vector scaling are both logit transformation techniques proposed by Guo et al. (2017) 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) (Kull et al., 2019): the term $\lambda(\frac{1}{m(m-1)} \sum_{i \neq j} w_{ij}^2) + \mu(\frac{1}{m} \sum_j b_j^2)$ is added to the training loss, where λ and μ are the regularization hyperparameters.

Dirichlet calibration Dirichlet calibration is a method proposed by Kull et al. (2019) 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 ln is a vectorized natural-logarithm function. Similar to matrix scaling, Dirichlet calibration is also trained with ODIR to prevent overfitting.

Intra order-preserving functions Rahimi et al. (2020) 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 $argsort\, \hat{\boldsymbol{c}}(\boldsymbol{z}) = argsort\, \boldsymbol{z}$. In addition, the authors also showed that in practice it is better to fit a diagonal subfamily of IOP functions as they contain less parameters. 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 $\hat{\boldsymbol{c}}$. It can be noted that temperature scaling uses a subfamily of diagonal IOP functions: it uses linear diagonal IOP functions where the bias term equals 0.

Decision calibration Zhao et al. (2021) proposed a non-parametric calibration method for the context of decision making settings. The method works iteratively by partitioning the probability simplex and applying an adjustment on each partition. The partition and adjustment are both determined by the average difference between the label and prediction values.

Gaussian process calibration Wenger et al. (2020) proposed a natively multi-class non-parametric calibration method that uses a latent Gaussian process to learn the calibrating transformation. The method applies the same transformation for all the classes.

3 Contributions

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

1. The work formulates two assumptions about the true calibration map that have been previously used but not clearly stated in the calibration literature.
2. By explicitly acknowledging the assumptions, we propose an intuitive and simple non-parametric post-hoc calibration method.

3.1 Proposed assumptions

Before introducing the proposed assumptions, a small introduction is needed. According to the definition of multi-class calibration given in Section 2.2, a prediction vector $\hat{\boldsymbol{p}}$ is considered calibrated if $\hat{\boldsymbol{p}} = \mathbb{E}\left[\boldsymbol{Y}|\hat{\boldsymbol{P}} = \hat{\boldsymbol{p}}\right]$. Therefore, a calibrating transformation of a prediction $\hat{\boldsymbol{p}}$ could be found if we had a good estimate of its true conditional class distribution $\mathbb{E}\left[\boldsymbol{Y}|\hat{\boldsymbol{P}} = \hat{\boldsymbol{p}}\right]$ — we could simply set the prediction value $\hat{\boldsymbol{p}}$ equal to the estimate. Similarly, if we were to approach calibration from the definition of calibration error $CE(\hat{\boldsymbol{p}}) = \hat{\boldsymbol{p}} - \mathbb{E}\left[\boldsymbol{Y}|\hat{\boldsymbol{P}} = \hat{\boldsymbol{p}}\right]$ given in Section 2.3, it would suffice for calibration if we had a good calibration error estimate — we could subtract the estimate from the prediction and our output would be calibrated.

One obvious weak estimator for the true class distribution could be the (single) label value $\boldsymbol{Y}$ corresponding to $\hat{\boldsymbol{p}}$. The estimator would clearly be unbiased as for each $\hat{\boldsymbol{p}}$, the average value of $\boldsymbol{Y}$ is equal to $\mathbb{E}[\boldsymbol{Y}|\hat{\boldsymbol{P}} = \hat{\boldsymbol{p}}]$, and hence, $\mathbb{E}[\boldsymbol{Y}|\hat{\boldsymbol{P}} = \hat{\boldsymbol{p}}] - \mathbb{E}[\boldsymbol{Y}|\hat{\boldsymbol{P}} = \hat{\boldsymbol{p}}] = 0$. However, this estimator $\boldsymbol{Y}$ would have very high variance as it is based on a sample with just one element. Likewise, an unbiased high variance estimator $\widehat{CE}$ could be constructed for the calibration error as the difference between the prediction and its label $\widehat{CE}(\hat{\boldsymbol{p}}) = \hat{\boldsymbol{p}} - \boldsymbol{Y}$. Unfortunately, both of these simple estimators have too high variance to be used for calibration. However, if we made some assumptions about the calibration map of our classifier, we could construct good estimators that make use of these weaker estimators.

Assumption of locally equal calibration errors (LECE) We propose to assume that 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 validation data set, the LECE assumption allows us to construct a considerably better estimator $\widehat{CE}_{neigh}$ for the calibration error of prediction $\hat{\boldsymbol{p}}$ than the previously introduced weak estimator $\widehat{CE}(\hat{\boldsymbol{p}}) = \hat{\boldsymbol{p}} - \boldsymbol{Y}$. First, we need to find a close neighborhood of $\hat{\boldsymbol{p}}$ where the assumption holds. A stronger estimator can then be constructed if we average across the weak calibration error estimator values of all validation set data points belonging to that close neighborhood

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

where the predictions $\hat{\boldsymbol{p}}_1, \ldots, \hat{\boldsymbol{p}}_k$ with labels $\boldsymbol{Y}_1, \ldots, \boldsymbol{Y}_k$ are the validation set data points belonging the close neighborhood of $\hat{\boldsymbol{p}}$.

Thanks to our assumption, the estimator $\widehat{CE}_{neigh}$ remains approximately unbiased as demonstrated by the following chain of equations where all expectations are to be taken conditioned on $\hat{\boldsymbol{P}}_1 = \hat{\boldsymbol{p}}_1, \ldots, \hat{\boldsymbol{P}}_k = \hat{\boldsymbol{p}}_k$:

$$\begin{aligned}
\mathbb{E}\left[\widehat{CE}_{neigh}(\hat{\boldsymbol{p}})\right] - CE(\hat{\boldsymbol{p}}) &= \frac{1}{k}\sum_{i=1}^{k}\mathbb{E}\left[(\hat{\boldsymbol{p}}_i - \boldsymbol{Y}_i)\right] - CE(\hat{\boldsymbol{p}}) \\
&= \frac{1}{k}\sum_{i=1}^{k}\left(\hat{\boldsymbol{p}}_i - \mathbb{E}\left[\boldsymbol{Y}_i\right]\right) - CE(\hat{\boldsymbol{p}}) \\
&= \frac{1}{k}\sum_{i=1}^{k}CE(\hat{\boldsymbol{p}}_i) - CE(\hat{\boldsymbol{p}}) \approx \frac{1}{k}\sum_{i=1}^{k}CE(\hat{\boldsymbol{p}}) - CE(\hat{\boldsymbol{p}}) = 0.
\end{aligned}$$

The variance of the estimator decreases approximately linearly as the number of neighbors increases

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

Given the estimate $\widehat{CE}_{neigh}(\hat{\boldsymbol{p}})$, a calibrated prediction can finally 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 locally equal class distributions (LECD) 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 one-hot label vector 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, consider Figure 1. It depicts the calibration maps learned by different calibration methods on a 3-class synthetic calibration task. The exact details about the data set and applied methods are given in Section 4.1. Both the background colors and black arrows depict the learned transformation. The same information that is represented by the arrow is also represented by the RGB color value which is in a linear relation with the calibration error vector at that point: red for class 1, green for class 2, and blue for class 3. Figure 1f depicts

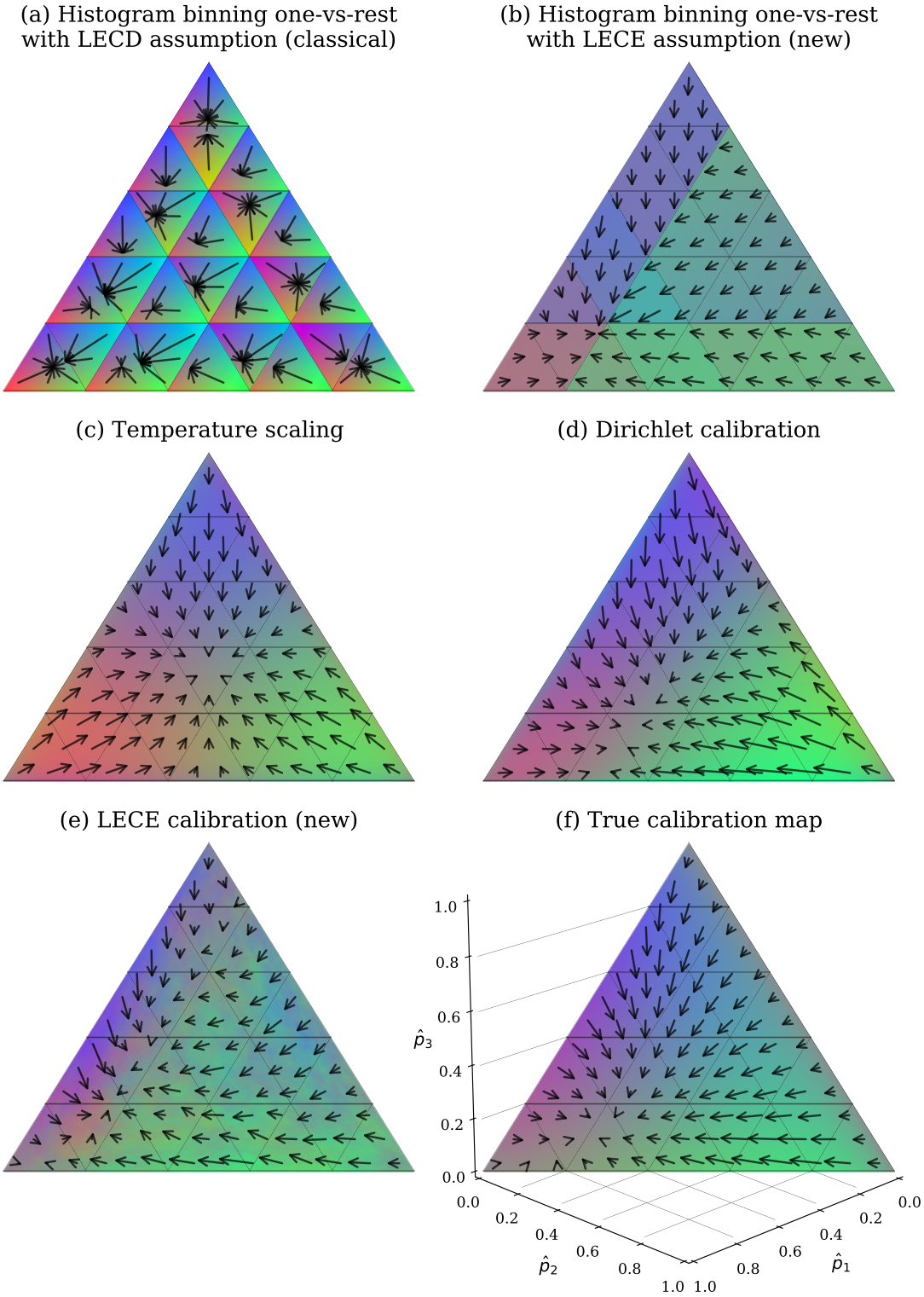

Figure 1: Illustrative example of calibration maps produced by different post-hoc calibration methods on a synthetic calibration task (a, b, c, d, e). The goal of the methods is to learn the true calibration map (f). Note the difference in histogram binning when applied with the LECD assumption (a) versus when applied with the LECE assumption (b): for LECD the black calibration arrows point to the same location in one bin; for LECE the arrows are the same in one bin. For details about the data set and applied methods, see Section 4.1.

the true calibration map which the different methods are trying to learn. Note that the true calibration map is something we never know in practice and here it has been manually created for the purpose of a synthetic calibration task to allow for visual comparison between different calibration methods. Figure 1a and Figure 1b depict the result of histogram binning methods applied in one-vs-rest style. Both of the binning methods have been applied with 5 equal-width bins, the only difference between the methods being the underlying assumption used. Classical histogram binning in Figure 1a uses the LECD assumption. The novel alternative in Figure 1b uses the LECE assumption. With the classical assumption, the end point of every calibrating arrow is the same in one bin. With the novel assumption, the calibrating arrow itself is the same in one bin. When comparing Figure 1a and Figure 1b to Figure 1f, histogram binning applied with the LECE assumption provides a closer imitation of the true calibration map than histogram binning applied with the LECD assumption. The results in Section 4.1 show that histogram binning with LECE assumption is better than with LECD by numeric metrics as well. The LECE assumption outperformed the LECD assumption on the real experiments also (as shown in Section 4.2.4) and is therefore preferred in this work.

Relation to prior work Neither of the two assumptions are completely novel as estimators based on them have been used in previous work. However, the assumptions themselves have never been explicitly stated. An estimator based on 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 widely used ECE calculation algorithm (Naeini et al., 2015) 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 similar to 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 (Zhao et al., 2021) 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.

An estimator based on the assumption of equal class distributions has also been previously used. It is the basis of the histogram binning method (Zadrozny & Elkan, 2001) where the close neighborhood is defined by the binning scheme; in each bin, the average one-hot label vector is used to calibrate the predictions. The weighted average one-hot label vector is also used in recent work by Popordanoska et al. (2021) 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 vectors is then used to estimate the calibrated prediction.

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

1. How many instances should the close neighborhood cover?
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 simple 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 (Zhao et al., 2021) defines the close neighborhoods by a linear partition that splits the probability simplex such that the calibration error estimates would be maximized. Popordanoska et al. (2021) define the neighborhood through assigning weights with a kernel function.

3.2 LECE calibration

As for the multi-class case no natural binning scheme exists, an alternative is required. One intuitive approach would be to define the close neighborhood separately for every data point: for some prediction

$\hat{\boldsymbol{p}}$, the close neighborhood could be defined by the k closest instances on the validation data set. Defining the neighborhood this way results basically in a modified k-nearest-neighbors algorithm that we call LECE calibration. For any uncalibrated prediction $\hat{\boldsymbol{p}}$, LECE calibration would

1. Find the k closest predictions on the validation data set $\hat{\boldsymbol{p}}_1, \ldots, \hat{\boldsymbol{p}}_k$ according to some distance function d.
2. Estimate the calibration error of $\hat{\boldsymbol{p}}$ by finding the average difference between its neighbors' prediction and label $\widehat{CE}_{neigh}(\hat{\boldsymbol{p}}) = \frac{1}{k}\sum_{i=1}^{k}(\hat{\boldsymbol{p}}_i - \boldsymbol{y}_i) = \frac{1}{k}\sum_{i=1}^{k}\widehat{CE}(\hat{\boldsymbol{p}}_i)$.
3. Subtract the calibration error estimate $\widehat{CE}_{neigh}(\hat{\boldsymbol{p}})$ from the uncalibrated prediction $\hat{\boldsymbol{p}}$ to calibrate it $\hat{\boldsymbol{c}}(\hat{\boldsymbol{p}}) = \hat{\boldsymbol{p}} - \widehat{CE}_{neigh}(\hat{\boldsymbol{p}})$.

Similary to the LECE calibration algorithm, a LECD calibration algorithm could be defined as well, with the only difference being in the underlying assumption used — instead of steps 2 and 3 of the algorithm, the method would instead set the calibrated prediction equal to the average label value of the neighbors $\hat{\boldsymbol{c}}(\hat{\boldsymbol{p}}) = \frac{1}{k}\sum_{i=1}^{k}\boldsymbol{y}_i$.

For the distance function d, we considered Kullback-Leibler divergence and Euclidean distance. As shown in the real data experiments in Section 4.2.4, Kullback-Leibler divergence performed slightly better. To find the neighbors of $\hat{\boldsymbol{p}}$, Kullback-Leibler divergence is applied as $d_{kl}(\hat{\boldsymbol{p}}, \cdot)$ where

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

the term in the sum is considered 0 if $\hat{p}_{1i} = 0$ and ∞ if $\hat{p}_{1i} \neq 0$ and $\hat{p}_{2i} = 0$, and m is the number of classes (Murphy, 2022).

Thresholding tiny probabilities A problem inherent to the non-parametric LECE (and LECD) calibration method is its inability to work well for tiny probabilities. This is because the method uses an estimator, which has some built in errors coming from its bias and/or variance. For class probabilities that are very near to 0, these errors of the estimator become very large proportionally to the probability. This could even lead to situations, where the LECE method produces output that is smaller than 0 for some classes and could no longer be interpreted as probabilities. Therefore, to overcome this problem with small probabilities, we opted to introduce one more parameter to the calibration method: a threshold value $t \in \mathbb{R}$ which sets a lower limit when to apply the method. For any class probability smaller than t we do not apply the method. In addition to that, for any output calibrated class probability smaller than t we undo the calibrating transformation and keep the original predicted value for this class. More precisely, given the prediction vector $\hat{\boldsymbol{p}}$, and the would-be calibrated prediction vector $\hat{\boldsymbol{c}}(\hat{\boldsymbol{p}}) = \hat{\boldsymbol{p}} - \widehat{CE}_{neigh}(\hat{\boldsymbol{p}})$, if for the i-th class $\hat{p}_i \leq t$ or $\hat{c}(\hat{\boldsymbol{p}})_i \leq t$, then we set $\hat{c}(\hat{\boldsymbol{p}})_i = \hat{p}_i$. Thresholding can cause the final output to no longer sum to 1, so to solve this, as a final step we divide the output 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 (Zhao et al., 2021). First, temperature scaling learns the calibration map on the validation data set and then their method fine-tunes the result of temperature scaling using the temperature scaled validation data. The benefit of composing parametric and non-parametric calibration methods was also shown by Zhang et al. (2020) who noted that isotonic regression applied in a one-versus-rest manner works better if it is applied on top of temperature scaling. A similar observation is true for the proposed non-parametric LECE calibration method in this work as well. This is expected because the violations of the LECE assumptions are likely to diminish when the calibration errors become smaller, which is the case when applying temperature scaling.The experiments on real data in Section 4.2 show that the proposed method loses to existing parametric post-hoc calibration methods when applied directly, but wins when applied on top of temperature scaling.

To summarize, the LECE calibration method involves thresholding of tiny probabilities and it works best when composed with a parametric method.

4 Experiments

The experiments' section consists of two parts:

- 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 method with its competitors and see if it can offer improvement over the state-of-the-art.

4.1 Synthetic data experiment

To illustrate the differences between the assumption of locally equal calibration errors (LECE) and the assumption of locally equal class distributions (LECD), and the limitations of the existing post-hoc calibration methods, a synthetic data set was created. A synthetic approach allows to define the 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.

Table 1: Results of the synthetic experiment.

		ours			ours	
	H-LECD	H-LECE	TS	DIR	LECE	true
confidence CE	0.056_5	0.019_2	0.03_4	0.022_3	$\mathbf{0.017_1}$	0.000
classwise CE	0.049_5	0.016_2	0.027_4	0.019_3	$\mathbf{0.015_1}$	0.000
Brier score	0.446_5	$\mathbf{0.436_1}$	0.439_4	0.437_3	$\mathbf{0.436_1}$	0.435
log-loss	0.769_5	0.745_2	0.750_4	0.745_2	$\mathbf{0.739_1}$	0.735
accuracy	0.671_4	0.672_3	0.671_4	$\mathbf{0.673_1}$	$\mathbf{0.673_1}$	0.673
average rank	4.8	2.0	4.0	2.4	1.0	

4.1.1 Data generation

For the experiment, a 3-class validation data set consisting of 5000 prediction and label pairs, and a test data set consisting of 10000 prediction and label pairs were generated. Note that as calibration applies on the predictions and does not depend on the original feature space, we are directly generating predictions without considering the features nor the classification model at all. 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 is defined as $\boldsymbol{c}(\hat{\boldsymbol{p}}) = (\hat{p}_1^{0.8} + \frac{\hat{p}_1 \cdot \hat{p}_2}{5}, \hat{p}_2 + \frac{\hat{p}_1 \cdot \hat{p}_3}{3}, \hat{p}_3 + \frac{\hat{p}_1 \cdot \hat{p}_2}{10})/Z$, where Z is the renormalizing term to sum to 1. The chosen function is depicted in Figure 1f. Note that the results of the synthetic experiment should be considered purely illustrative as the function $\boldsymbol{c}(\cdot)$ was chosen rather arbitrarily.

4.1.2 Compared post-hoc calibration methods

On the synthetic task several post-hoc calibration methods were evaluated:

- Two different histogram binnings with 5 equal-width bins: a classical version with the LECD assumption (**H-LECD**) and a novel version with the LECE assumption (**H-LECE**). The histogram binning methods were chosen to visualise the difference between the two assumptions formalized in Section 3.
- Temperature scaling (**TS**) to illustrate the limitations of symmetric calibration methods which can only learn transformations that act the same way for all the classes. Temperature scaling was applied to $log(\hat{\boldsymbol{p}})$ as no logits were available for the task.

- Dirichlet calibration (**DIR**) to show the limited expressivity of the otherwise powerful parametric methods. Dirichlet calibration was fit without regularization as the number of parameters is low with 3 classes.
- Our proposed LECE calibration method (**LECE**) to illustrate its merits. The method was applied with $k = 500$ neighbors and threshold value $t = 0$ (thus applying thresholding only to ensure that outputs are non-negative).

4.1.3 Results

The learned calibration maps of the methods are depicted in Figure 1. Table 1 contains the results for the synthetic experiments; it shows the average of 10 data generation seeds. 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. For synthetic experiments, the ECE measures are replaced with the true calibration errors (CE) as the true calibration function is known (see Section 2.3 for details): CE is measured with $\alpha = 1$ and the expectation is replaced with empirical test set average.

First, when comparing the two histogram binnings, it can be seen that the evaluation metrics confirm the visual intuition available from Figure 1a and Figure 1b: the new proposed LECE assumption significantly outperforms the classical LECD assumption. Second, as seen in Figure 1c, temperature scaling is clearly not sufficient for the task as it is limited to only a symmetrical transformation. Third, Dirichlet calibration, one of the strongest existing competitors learns a result close to the true calibration map, but is held back by its parametric family. Overall, the proposed LECE calibration method depicted in Figure 1e learns the most similar transformation to the true calibration map.

4.2 Real data experiments

The goal of the real data experiments is to see if the proposed method can improve state-of-the-art in practical settings.

4.2.1 Data sets and models

CIFAR-10 and CIFAR-100 data sets (Krizhevsky, 2009) are used for the experiments. On both of the data sets, the predictions of convolutional neural networks ResNet-110 (He et al., 2016), ResNet Wide 32 (Zagoruyko & Komodakis, 2016), DenseNet-40 (Huang et al., 2017) are used. The logits of these three CNN-s on the calibration and test set were the same as used by Kull et al. (2019) and Rahimi et al. (2020). An overview of the used data sets, classifiers, and data set sizes is given in Table 2.

4.2.2 Compared post-hoc calibration methods

On the real data sets the following methods are compared:

- Uncalibrated predictions (**uncal**) to have a baseline.

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

			Data set size		
Data set	Models	Classes	Training	Validation	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

Table 3: Confidence ECE ($\times 10^2$).

			ours						ours
		uncal	LECE	MS	IOP	GP	TS	TS+DEC	TS+LECE
C-10	DenseNet-40	5.49_8	3.72_7	0.91_4	0.80_2	0.86_3	0.92_5	1.13_6	$\mathbf{0.69_1}$
	ResNet-110	4.75_8	3.31_7	0.99_5	0.91_3	0.88_2	0.94_4	1.01_6	$\mathbf{0.65_1}$
	ResNet Wide 32	4.48_8	2.05_7	0.75_5	0.69_3	$\mathbf{0.42_1}$	0.69_3	0.76_6	0.60_2
	average rank	8.0	7.0	4.7	2.7	2.0	4.0	6.0	1.3
C-100	DenseNet-40	21.16_8	10.49_7	1.22_4	3.45_5	0.92_2	$\mathbf{0.79_1}$	3.85_6	1.18_3
	ResNet-110	18.48_8	6.36_7	2.31_4	2.79_5	$\mathbf{1.81_1}$	2.13_3	3.40_6	1.99_2
	ResNet Wide 32	18.78_8	15.53_7	1.85_5	1.03_3	0.86_2	1.41_4	3.34_6	$\mathbf{0.85_1}$
	average rank	8.0	7.0	4.3	4.3	1.7	2.7	6.0	2.0

Table 4: Classwise ECE ($\times 10^2$).

			ours						ours
		uncal	LECE	MS	IOP	GP	TS	TS+DEC	TS+LECE
C-10	DenseNet-40	0.445_8	0.411_7	$\mathbf{0.214_1}$	0.251_3	0.259_5	0.255_4	0.291_6	0.234_2
	ResNet-110	0.358_8	0.297_7	$\mathbf{0.180_1}$	0.212_2	0.212_2	0.216_4	0.248_6	0.216_4
	ResNet Wide 32	0.496_8	0.292_4	$\mathbf{0.181_1}$	0.452_7	0.438_5	0.446_6	0.282_3	0.246_2
	average rank	8.0	6.0	1.0	4.0	4.0	4.7	5.0	2.7
C-100	DenseNet-40	0.167_7	0.251_8	0.095_2	0.110_5	0.102_3	0.102_3	0.139_6	$\mathbf{0.094_1}$
	ResNet-110	0.132_6	0.159_8	0.110_5	0.096_4	$\mathbf{0.092_1}$	0.094_3	0.135_7	$\mathbf{0.092_1}$
	ResNet Wide 32	0.125_7	0.137_8	0.094_2	0.105_4	0.105_4	0.103_3	0.124_6	$\mathbf{0.089_1}$
	average rank	6.7	8.0	3.0	4.3	2.7	3.0	6.3	1.0

- Matrix scaling with ODIR (**MS**); best hyperparameters for the data set and classifier combinations were provided by the authors of ODIR for matrix scaling (Kull et al., 2019).
- Diagonal subfamily of intra-order preserving functions (**IOP**); best hyperparameters for the data set and classifier combinations were obtained from the original article (Rahimi et al., 2020).
- Gaussian process calibration (**GP**) applied on logits (Wenger et al., 2020).
- Temperature scaling (**TS**) (Guo et al., 2017).
- Decision calibration (Zhao et al., 2021); trained to achieve decision calibration with respect to all loss functions with 2 decisions; number of trained iterations was determined by looking at the test set (!) Brier score to save computational resources, thus giving a slight unfair advantage to this method; the final output was normalised to sum to 1 which was not done in the original implementation but is inevitable for log-loss evaluation; applied in composition with temperature scaling (**TS+DEC**) as recommended by the original paper.
- Our proposed method; optimal hyperparameters were found with 10-fold cross-validation grid search optimized by log-loss from neighborhood size proportions q of the training data set $q = k/n \in \{0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.1, 0.2, 1.0\}$, and threshold values $t \in \{0, 0.00125, 0.0025, 0.005, 0.01, 0.02, 0.04, 0.05, 0.10, 1.0\}$. Note that including $t = 1$ as a possible value allows the calibration method to learn the identity map. Neighborhood size proportion 0.01 stands for $0.01 \cdot 5000 = 50$ neighbors in the validation set and $0.01 \cdot 4500 = 45$ neighbors in one fold of the 10-fold cross-validation as there are 5000 total data points in the validation set. The reason

for using a fixed proportion q instead of a fixed number of neighbours k is to ensure that the neighborhood would cover approximately the same subregion of the probability simplex on data folds of different sizes. LECE calibration is applied in composition with temperature scaling (**TS+LECE**) and without temperature scaling (**LECE**).

4.2.3 Results

In the following paragraphs the results for confidence ECE, classwise ECE, and log-loss are presented.

Confidence ECE Table 3 presents the results for confidence ECE. Both confidence and classwise ECE are measured with 15 equal-sized bins or also known as data dependent bins. According to confidence ECE, the best performing methods are GP and our proposed method TS+LECE. Without TS, LECE performs poorly on the real data sets: it is heavily outperformed by every other method. However, when applied in composition with TS, the result of TS is improved for 5 out of 6 cases.

Table 5: Log-loss.

			ours						ours
		uncal	LECE	MS	IOP	GP	TS	TS+DEC	TS+LECE
C-10	DenseNet-40	0.428_8	0.310_7	$\mathbf{0.222_1}$	0.225_3	0.226_5	0.225_3	0.266_6	0.223_2
	ResNet-110	0.358_8	0.267_7	$\mathbf{0.204_1}$	0.208_4	0.206_2	0.209_5	0.232_6	0.206_2
	ResNet Wide 32	0.382_8	0.264_7	$\mathbf{0.182_1}$	0.192_5	0.191_3	0.191_3	0.234_6	0.185_2
	average rank	8.0	7.0	1.0	4.0	3.3	3.7	6.0	2.0
C-100	DenseNet-40	2.017_8	1.739_7	$\mathbf{1.047_1}$	1.067_5	1.056_3	1.057_4	1.338_6	1.054_2
	ResNet-110	1.694_8	1.498_7	$\mathbf{1.073_1}$	1.106_5	1.082_2	1.092_4	1.453_6	1.085_3
	ResNet Wide 32	1.802_8	1.576_7	$\mathbf{0.931_1}$	0.945_4	0.939_3	0.945_4	1.284_6	0.937_2
	average rank	8.0	7.0	1.0	4.7	2.7	4.0	6.0	2.3

Classwise ECE Table 4 presents the results for classwise ECE. 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 (Nixon et al., 2019). On CIFAR-10, matrix scaling is clearly the best performing method. On CIFAR-100 our proposed TS+LECE performs best but only marginally: many other methods offer similar scores. The results on CIFAR-10 ResNet Wide 32 expose the limitations of symmetrical methods that perform the same transformation for all the classes. For that case, IOP, GP, and TS all fail and produce bad classwise ECE scores around 0.440 which is only slightly lower than the uncalibrated result 0.496. Methods not limited by symmetry offer substantially better results, all producing scores less or equal to 0.300 with matrix scaling even reaching as low as 0.181. Without TS, LECE performs again poorly: for CIFAR-100 it even worsens the result of uncalibrated predictions. However, similarly to confidence ECE, when applied in composition with TS, it offers consistent improvements over TS.

Log-loss Table 5 displays the results for log-loss. The best method according log-loss is clearly matrix scaling, being ranked first every time. The second best method is our proposed TS+LECE. The limitations of symmetrical methods can be seen according to log-loss as well: on CIFAR-10 ResNet Wide 32, IOP, GP, and TS perform worse than MS and TS+LECE. Without TS, LECE again performs poorly, but in composition with TS, it consistently offers improvements over TS.

4.2.4 Ablation study

To understand the importance of the distance function used in the LECE calibration algorithm, and to compare LECE calibration with LECD calibration, we repeat the real data experiments for the following methods

Table 6: Ablation study, confidence ECE ($\times 10^2$).

		LECE	LECE$_{euc}$	LECD	TS+LECE	TS+LECE$_{euc}$	TS+LECD
C-10	DenseNet-40	3.72_6	3.30_4	3.43_5	$\mathbf{0.69_1}$	0.76_2	1.19_3
	ResNet-110	3.31_6	3.20_5	2.69_4	$\mathbf{0.65_1}$	0.66_2	0.94_3
	ResNet Wide 32	2.05_5	2.93_6	1.93_4	0.60_3	0.53_2	$\mathbf{0.50_1}$
	average rank	5.7	5.0	4.3	1.7	2.0	2.3
C-100	DenseNet-40	10.49_5	11.16_6	5.97_4	1.18_2	1.21_3	$\mathbf{0.79_1}$
	ResNet-110	6.36_5	12.02_6	3.53_4	1.99_2	2.03_3	$\mathbf{1.71_1}$
	ResNet Wide 32	15.53_6	14.21_5	5.76_4	$\mathbf{0.85_1}$	1.16_2	1.41_3
	average rank	5.3	5.7	4.0	1.7	2.7	1.7

Table 7: Ablation study, classwise ECE ($\times 10^2$).

		LECE	LECE$_{euc}$	LECD	TS+LECE	TS+LECE$_{euc}$	TS+LECD
C-10	DenseNet-40	0.411_4	0.670_6	0.453_5	$\mathbf{0.234_1}$	0.238_2	0.267_3
	ResNet-110	0.297_4	0.557_6	0.335_5	0.216_2	$\mathbf{0.202_1}$	0.216_2
	ResNet Wide 32	0.292_5	0.732_6	0.285_4	$\mathbf{0.246_1}$	0.274_2	0.277_3
	average rank	4.3	6.0	4.7	1.3	1.7	2.7
C-100	DenseNet-40	0.251_5	0.336_6	0.172_4	$\mathbf{0.094_1}$	0.096_2	0.102_3
	ResNet-110	0.159_5	0.269_6	0.123_4	$\mathbf{0.092_1}$	0.095_3	$\mathbf{0.092_1}$
	ResNet Wide 32	0.137_5	0.332_6	0.126_4	$\mathbf{0.089_1}$	0.095_2	0.103_3
	average rank	5.0	6.0	4.0	1.0	2.3	2.3

- LECE calibration with Euclidean distance instead of Kullback-Leibler divergence (**LECE**$_{euc}$ and **TS+LECE**$_{euc}$),

- LECD calibration as described in Section 3.2 — otherwise the same method as LECE calibration but using the LECD assumption instead of the LECE assumption (**LECD** and **TS+LECD**).

The method parameters were chosen with 10-fold cross-validation from the same parameter sets as for LECE calibration described in Section 4.2.2. Table 6 presents the results for confidence ECE, Table 7 the results for classwise ECE, and Table 8 the results for log-loss. The three tables are discussed in unison, as the methods are ranked similarly across the tables, and the key conclusions to be made from the tables are the same.

The best method on average in all the tables is TS+LECE. TS+LECE outperforms its derivative with Euclidean distance TS+LECE$_{euc}$ in almost all cases with a few exceptions: it loses once in confidence ECE, once in classwise ECE, and is tied twice in log-loss. TS+LECE outperforms TS+LECD as well: it performs better in 13 cases, is tied in 2, and loses in 3 out of the 18 total rows in the three tables.

When comparing the methods with and without TS, it can be seen that TS is crucial for all of them. Adding the composition with TS improves the result in all cases and by a very large margin. For cases without TS, LECE and LECE$_{euc}$ perform otherwise similarly but for classwise ECE the method LECE$_{euc}$ fails. Therefore, similary to the LECE methods applied in composition with TS, Kullback-Leibler divergence can be concluded to perform better than Euclidean distance for the non-compositional case as well. Comparing LECE with LECD, LECE seems to perform better for CIFAR-10 but LECD for CIFAR-100. Yet, as the compositional TS+LECE methods heavily outperformed the non-compositonal LECE methods, the final conclusion would still be that LECE assumption is better than the LECD assumption.

Table 8: Ablation study, log-loss.

		LECE	LECE$_{euc}$	LECD	TS+LECE	TS+LECE$_{euc}$	TS+LECD
C-10	DenseNet-40	0.310_5	0.305_4	0.319_6	$\mathbf{0.223_1}$	0.224_2	0.226_3
	ResNet-110	0.267_5	0.265_4	0.279_6	$\mathbf{0.206_1}$	$\mathbf{0.206_1}$	0.209_3
	ResNet Wide 32	0.264_5	0.260_4	0.272_6	$\mathbf{0.185_1}$	0.187_2	0.188_3
	average rank	5.0	4.0	6.0	1.0	1.7	3.0
C-100	DenseNet-40	1.739_6	1.681_5	1.437_4	$\mathbf{1.054_1}$	$\mathbf{1.054_1}$	1.057_3
	ResNet-110	1.498_5	1.516_6	1.306_4	$\mathbf{1.085_1}$	1.087_2	1.092_3
	ResNet Wide 32	1.576_6	1.544_5	1.272_4	$\mathbf{0.937_1}$	0.941_2	0.945_3
	average rank	5.7	5.3	4.0	1.0	1.7	3.0

Table 9: Optimal neighborhood proportion q chosen by cross-validation.

		LECE	LECE$_{euc}$	LECD	TS+LECE	TS+LECE$_{euc}$	TS+LECD
C-10	DenseNet-40	0.04	0.1	0.04	0.04	0.03	0.07
	ResNet-110	0.06	0.1	1.0	0.03	0.02	0.01
	ResNet Wide 32	0.02	0.1	0.02	0.01	0.01	0.01
C-100	DenseNet-40	0.05	0.03	1.0	0.01	0.02	0.01
	ResNet-110	0.01	0.05	1.0	0.04	0.03	0.01
	ResNet Wide 32	1.0	0.03	1.0	0.04	0.05	0.01

Table 9 shows the optimal neighborhood proportion parameter q chosen by the 10-fold cross-validation for the methods. Many different values are represented, starting from 0.01 corresponding to $0.01 \cdot 5000 = 50$ neighbors, up to 1.0, corresponding to the whole validation data set of 5000 neighbors. For TS+LECE, the neighborhood proportion remains in range between 0.01 to 0.04 corresponding to 50 to 200 neighbors.

Table 10 shows the optimal threshold parameter chosen by the 10-fold cross-validation. For methods applied without TS, 0 seems to be a good value as it was chosen by LECE and LECE$_{euc}$ in all cases. For methods applied in composition with TS, the chosen threshold values remain usually small: between 0.0025 and 0.02. TS+LECD is an exception as it uses larger threshold values. In three cases out of six it even picked $t = 1.0$, meaning it learned the identity map and kept the result of TS.

4.2.5 Discussion

Overall, from the experiments it can be concluded that while many strong methods exist, our proposed TS+LECE can still offer improvements. Symmetrical methods IOP, GP, and TS perform generally well,

Table 10: Optimal threshold t chosen by cross-validation.

		LECE	LECE$_{euc}$	LECD	TS+LECE	TS+LECE$_{euc}$	TS+LECD
C-10	DenseNet-40	0	0	0	0.0025	0.0025	0.1
	ResNet-110	0	0	0.05	0.01	0.01	1.0
	ResNet Wide 32	0	0	0	0.01	0.02	0.05
C-100	DenseNet-40	0	0	0.005	0.02	0.02	1.0
	ResNet-110	0	0	0.005	0.01	0.02	0.1
	ResNet Wide 32	0	0	0.005	0.005	0.005	1.0

but can fail for problems where asymmetrical transformations are needed as shown by classwise ECE and log-loss on CIFAR-10 ResNet Wide 32. Matrix scaling could be considered the best method according to classwise ECE and log-loss, but according to confidence ECE it is clearly outperformed by other methods. TS+LECE avoids the problems of symmetrical methods and offers the best confidence ECE, being second best in classwise ECE and log-loss.

Another aspcect to notice from the experiments, is the behaviour of LECE with the number of classes. On the synthetic task with 3 classes the method performs very well without TS. However, on the real tasks with 10 and 100 classes, the method fails unless it is used on top of TS. This is due to problems arising from the curse of dimensionality inherent to the proposed non-parametric approach. In higher dimensions, the probability space is more sparsely populated as every data point starts to become approximately equally distant from every other data point. Because of this, the neighborhood sizes grow and the applied assumption of locally equal calibration errors becomes very bold, lowering the effectiveness of the method. However, applying LECE on top of TS makes the assumption to be more realistic, as the calibration errors are smaller after TS and hence the errors of neighbors are more similar, as required by the LECE assumption.

5 Conclusion

This work explored the field of post-hoc calibration methods for multi-class classifiers. Two assumptions about the true calibration map were formalized that have been previously used for creating estimators, but despite this have never been clearly stated: assuming locally equal calibration errors (LECE) and assuming locally equal class distributions (LECD). Based on the more reasonable of the assumptions, a non-parametric calibration method was proposed — LECE calibration. The used assumption states that the calibration error of a data point can be modeled by the calibration errors of its neighbors. This 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.

Experiments were carried out on three convolutional neural networks trained on CIFAR-10 and CIFAR-100 to compare the proposed method with its competitors. The experimental results on real data showed that the proposed method alone is clearly not competitive for cases with many classes and a limited validation data set due to problems arising from the curse of dimensionality. However, when applying the proposed method in composition with temperature scaling, it tops the state-of-the-art in confidence ECE and is close to the best according to classwise ECE and log-loss.

For future work, the limitations of the proposed approach could be studied more thoroughly. How does improvement in calibration depend on the number of classes in the data set; on the validation data set size; or on the distribution of the predictions? In addition, the composition of different calibration methods could be studied further as this work and several previous (Zhang et al., 2020; Zhao et al., 2021) have shown the possible benefits.

References

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

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

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770–778, 2016.

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

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

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, pp. 68–85, 2015.

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, pp. 623–631, 2017.

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.

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

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

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

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.

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, pp. 2980–2988, 2017.

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.

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.

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

Kevin P. Murphy. Probabilistic Machine Learning: An introduction. MIT Press, 2022. URL `probml.ai`. Last visited 2022-06-10.

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

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

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

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

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 2022-06-10.

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.

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, pp. 2818–2826, 2016.

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

Jonathan Wenger, Hedvig Kjellström, and Rudolph Triebel. Non-Parametric Calibration for Classification. In International Conference on Artificial Intelligence and Statistics, pp. 178–190, 2020.

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

Bianca Zadrozny and Charles Elkan. Transforming Classifier Scores into Accurate Multiclass Probability Estimates. In Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 694–699, 2002.

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

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, pp. 11117–11128, 2020.

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.