Neural Computing and Applications
https://doi.org/10.1007/s00521-024-09505-4

ORIGINAL ARTICLE

Adaptive temperature scaling for Robust calibration of deep neural networks

Sergio A. Balanya[1] · Juan Maroñas[2,3] · Daniel Ramos[1]

Received: 14 March 2023 / Accepted: 14 January 2024

Abstract
In this paper, we study the post-hoc calibration of modern neural networks, a problem that has drawn a lot of attention in recent years. Despite the plethora of calibration methods proposed, there is no consensus yet on the inherent complexity of the task and, while some authors claim that simple functions solve the problem, others suggest that more expressive models are needed to capture misscalibration. As a first approach, we focus on the task of confidence scaling, specifically on post-hoc methods that generalize Temperature Scaling, which we refer to as the Adaptive Temperature Scaling family. We begin by demonstrating that while complex models like neural networks provide an advantage when there is ample data, they fail in scenarios where it is limited, notably common in fields like medical diagnosis. We then show how under this ideal data conditions the more expressive methods learn a relationship between the entropy of a prediction and its level of overconfidence, and based on this observation, we propose Entropy-based Temperature Scaling, a simple method that scales the confidence of a prediction according to this relationship. Results show that our method obtains state-of-the-art performance and is robust against data scarcity. Moreover, our proposed model enables a deeper understanding of the calibration process by the interpretation of the entropy as a measure of uncertainty in the network outputs.

Keywords Robust Calibration · Deep Neural Networks · Over-confidence · Uncertainty · Classification

1 Introduction

There is an increasing trend in using Deep Neural Networks (DNNs) to automate a multitude of tasks, like image classification for healthcare [1] and speech recognition [2] among many others. Some of these are high-risk applications in nature, for example, a False Negative given by a DNN used for cancer detection could be fatal for the patient. Hence it is of paramount importance to use reliable Machine Learning (ML) systems that acknowledge the uncertainty of their predictions. A probabilistic classifier that outputs a confidence value, or probability, for each class, allows to make Bayes decisions—i.e., optimum decisions leveraging the cost of such decisions [3]. So, by applying the Bayesian framework we can readily account for such uncertainty, assuming that the system produces *calibrated* confidence values.

The extent to which the confidence outputs of a classifier can be interpreted as class probabilities is what is known as the *calibration* of a classifier [4, 5]. Modern DNNs achieve very low test error rates but are not necessarily well-calibrated [6, 7]. Hence, there is a growing interest from the ML community toward improving the calibration of DNNs.

This work was written prior to Sergio A. Balanya joining Amazon.

✉ Sergio A. Balanya
sergio.alvarezb@estudiante.uam.es

Juan Maroñas
juan.maronnas@uam.es

Daniel Ramos
daniel.ramos@uam.es

1 AUDIAS Laboratory - Audio, Data Intelligence and Speech, Universidad Autónoma de Madrid, Calle Francisco Tomás y Valiente 11, 28049 Madrid, Spain

2 Machine Learning Group, Universidad Autónoma de Madrid, Calle Francisco Tomás y Valiente 11, 28049 Madrid, Spain

3 Cognizant Spain, Madrid, Spain

1.1 Related work

One approach to obtain better confidence estimates is to average the predictions of different models using ensembles [8] or taking a Bayesian approach to model learning [9, 10]. Data Augmentation techniques have also been used to improve calibration [11–13], as well as modified training objectives [14–16]. Among popular approaches, and the focus of this work, is the approach of post-hoc calibration, in which the predictions of an already trained classifier are re-calibrated (Fig. 1). Typically, other model, the calibrator, is trained on the outputs of the classifier to calibrate. This approach is very convenient since one can take advantage of the existing literature and use off-the-shelf ML systems that present all the desirable properties that make them so popular. Deep Learning models usually offer a good solution for any Machine Learning task and, for this reason, DNNs have become standard models with an easy application via public frameworks like Pytorch [17] and Tensorflow [18]. Post-hoc calibration allow us to use the existing stack to solve high-risk tasks and address over-confidence issues with little overhead.

Probably the most popular post-hoc calibration method for DNNs is Temperature Scaling (TS) proposed by [6]. It is a single parameter model that re-scales the confidence predictions by a *temperature factor*. The simplicity of this method and the fact that it seems to perform even better than more complex ones, led the authors to believe that the problem of re-calibration is inherently simple. However, recent alternatives based on expressive models like Bayesian Neural Networks [19] and Gaussian Processes [20] outperform TS, suggesting that re-calibration might be a more complex problem than it was previously assumed. On the other hand, expressive models can be more data-hungry and may require careful tuning when the amount of data is limited. Hence, TS still represents the standard calibration method as it yields a good trade-off between simplicity and performance, especially when recalibration data is limited.

Based on the observation that miscalibration on modern DNNs is often caused by over-confidence [6, 21], recent work proposes to learn more complex calibration functions than TS but from a constrained space by imposing some restrictions, like being accuracy-preserving [22] and order-invariant [23], inducing a bias toward the desired calibration functions. This approach shows promising results, but it may still fail in low-data scenarios, especially when using over-parameterized models. This can be a huge limitation in tasks where data for calibration is usually scarce, like certain language recognition tasks where some languages can be underrepresented [24, 25], or in the medical diagnosis of very rare diseases [26]. Thus, there is a need for calibration methods with low data requirements in many real applications.

We propose to use a simple model with a suitable inductive bias for the following reasons: First, the set of possible calibration functions that the model can learn—i.e., the hypothesis space—is reduced. This translates into an easier training objective requiring less tuning.

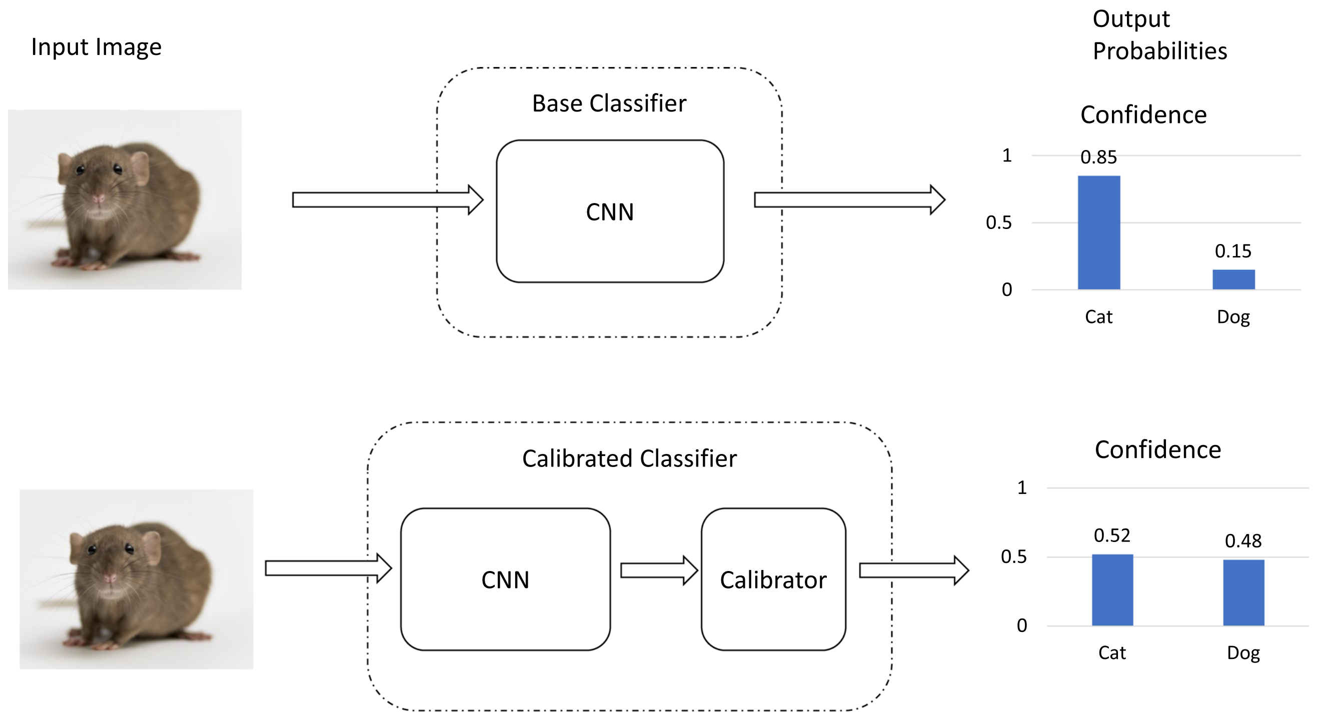


Fig. 1 Post-hoc calibration

Moreover, if the bias is well-specified, the learned calibration function will be more robust against a lack of training data, and will better generalize to other data [27]. The quality of the inductive bias depends on the knowledge we have of the task at hand. For example, the specific architecture of Convolutional Neural Networks (CNNs), based on convolution filters, explains their success on visual recognition tasks [28], even though by sharing weights the total number of parameters is reduced, constraining the learning space.

1.2 Contributions

To gain knowledge about the calibration of modern DNNs, we provide a study of post-hoc calibration methods. We analyze several state-of-the-art calibrators of varying degrees of expressiveness and robustness to help design models more resilient to data scarcity. We focus on the problem of confidence scaling as the bad calibration properties of DNNs are mainly attributed to over-confidence [6].

To perform this study we focus on Adaptive Temperature Scaling (ATS) methods, a family of calibration maps that generalizes TS by making the *temperature factor* input-dependent. This idea has already been proposed before in the literature [29], but the ATS method differs from those in the specific input-dependency. Previous methods propose to estimate the temperature factor as a function of the classifier input. ATS models, on the other hand, compute temperature factors directly from the output of the classifier via a *temperature function*. Within this family, we can compare several calibration methods which extend the expressiveness of TS in different ways.

We analyze and benchmark several calibration models focusing on the shape of those temperature functions that lead to better calibration. Results show that highly parameterized methods achieve high performance when there is plenty of data, but also that these are doomed to failure in low-data scenarios. By examining the behavior of expressive methods on ideal conditions, we notice that their *temperature function* shows a dependency between the entropy of a prediction and its degree of overconfidence. Based on this gained knowledge about the post-hoc calibration task, we develop Entropy-based Temperature Scaling (HTS), a method with a strong inductive bias that is robust to the size of the dataset and provides comparable performance to other state-of-the-art methods. We provide an interpretation of the method based on the use of entropy as a measure of uncertainty. This interpretation helps to understand the relation between overconfidence and uncertainty estimation.

The rest of the paper is organized as follows. First, we introduce some theoretical background of the calibration task. In Sect. 3 we propose some post-hoc calibration methods motivating their design, and also describe other existing techniques to which we compare our methods. Next, in Sect. 4 we describe the performed experiments, point out some observations, and show the results. Finally, in the last section, we give our conclusions and comment on possible future work.

2 Background

In this work we focus on the calibration of classifiers, so before delving into the concept of calibration we introduce the following notation to describe the multi-class classification task. Let $x \sim X \in \mathcal{X}$ be the input random variable with associated target $y \sim Y \in \mathcal{Y}$, where $y = [y_1, y_2, .., y_K]$ is a one-hot encoded label. The goal is to obtain a probabilistic model f for the conditional distribution $P(Y|X = x)$. Notice that, since Y is a categorical vector encoding the true class, any distribution $P(Y)$ on Y follows a categorical distribution. The model defines the function $f(x) = z$, $x \in \mathcal{X}$, $z \in \mathbb{R}^K$. The outputs z of the model are known as *logits* since they are later mapped to probability vectors via the softmax function:

$$q = \sigma_{\mathrm{SM}}(z) = \frac{\exp z}{\sum_{k=1}^{K} \exp z_k}, \tag{1}$$

where the exponential in the numerator is applied element-wise. The output $q \in \mathbb{S}^K$ is the corresponding probability vector that lies in the probability simplex in K classes $\mathbb{S}^K$ and q_k is the model predicted value for $P(y_k|x)$.

In practice there is no distribution $P(X, Y)$ (or we do not have access to it). Instead, we have a labeled data set $\mathcal{D}$ of N pair-realizations $\mathcal{D} = \{x^{(i)}, y^{(i)}\}_{i=1}^{N}$ that is used to approximate it. For example, DNNs are normally trained by minimizing the expected value of some loss function $\mathcal{L}(f(x), y)$, over the empirical distribution induced by placing a Dirac's *delta* at each point $\mathcal{D}$:

$$\sum_{(x,y)\in\mathcal{D}} \mathcal{L}(f(x), y).$$

2.1 Calibration

A probabilistic classifier is said to be *well-calibrated* whenever its confidence predictions for a given class match the chances of that class being the correct one [4, 5]. We can express this property as an equation in terms of the probability distributions introduced earlier:

$$P(y \mid q) = q, \ \forall q \in \mathbb{S}^K, \tag{2}$$

where $P(y \mid q)$ represents the relative class frequency —i.e., the proportion of each class on the set of all possible realizations of x for which the classifier predicts q.

From this expression, it is easy to derive a measure of miscalibration or *Calibration Error* (CE):

$$\mathrm{CE} = \mathbb{E}_{P(X,Y)}\left[\|P(y \mid q) - q\|_d\right]. \tag{3}$$

This is, the expected value of the d-norm of the difference between prediction vectors and the relative class proportions.

While this equation might be useful to illustrate the concept of miscalibration, it does not provide a feasible way to measure it. First, we cannot compute the expected value w.r.t. the non-existent distribution $P(X, Y)$, and, more importantly, there is no simple way of evaluating $P(y \mid q)$. The former can be readily solved by using the labeled dataset $\mathcal{D}$ as MC samples, yet the later is still a main limitation. Thus, further approximations are required to estimate the miscalibration of a classifier.

2.1.1 ECE

The most popular metric used to estimate the Calibration Error is the *Expected Calibration Error* (ECE) [6, 30]. This metric uses a histogram approach to model $P(y \mid q)$ and considers only top-label predictions—i.e., $\max(q)$. The samples of a given evaluation set $\mathcal{D}_{\text{test}}$ are partitioned into M bins $B_1, B_2, \ldots, B_M$ according to the confidence of their top prediction:

$$B_i := \left\{(x, y) \in \mathcal{D}_{\text{test}} : \frac{i-1}{M} < \max(q) \leq \frac{i}{M}\right\}.$$

Then the ECE is computed as:

$$\mathrm{ECE} = \sum_{i=1}^{M} \frac{|B_i|}{|\mathcal{D}_{\text{test}}|} \left|\mathrm{acc}(B_i) - \mathrm{cof}(B_i)\right|, \tag{4}$$

where $|\cdot|$ denotes the number of samples in a set, $\mathrm{acc}(B_i)$ is the accuracy of the classifier evaluated only on B_i, and $\mathrm{cof}(B_i)$ is the mean confidence of the top-label predictions in B_i.

Despite its popularity, this estimator provides unreliable results as it is biased and noisy [31–33]. Many improvements over the ECE have been proposed to mitigate these problems such as class-wise ECE and using variable confidence intervals [33]. However, there is not any binning scheme consistently reliable [34]. Anyway, ECE remains the most popular metric used by the community to measure miscalibration and we use it in our experiments to report results for the sake of comparison.

2.1.2 Proper scoring rules

One way to implicitly measure calibration is to use Proper Scoring Rules (PSRs). Any PSR can be decomposed into the sum of two terms [35], a *refinement* term and the so-called *reliability* or *calibration*. Thus, when evaluating the goodness of a classifier with a Proper Scoring Rule, one is also indirectly measuring calibration. The fact that the calibration component cannot be evaluated in isolation is what drives the community to use approximated metrics like ECE. Moreover, different PSRs may rank differently the same set of systems evaluated on the same data. Nevertheless, PSRs provide a theoretically grounded way of measuring the goodness of a classifier. Throughout this work, we use two different PSRs to evaluate models, the log-score or Negative Log-Likelihood (NLL) and the Brier score, both of them well-known [36].

2.1.3 Entropy

Because the output of a probabilistic classifier represents a categorical distribution, the entropy of a prediction vector q is defined as:

$$H(q) = -\sum_{k=1}^{K} q_k \log q_k. \tag{5}$$

It is easy to check that entropy reaches its maximum of $H(q) = -\log 1/K = \log K$ at $q = [1/K, 1/K, \ldots, 1/K]$. Likewise, it is minimized at the vertices of the probability simplex—i.e., $q_k = 1$ for any class k—where it takes a value $H(q) = 0$. From this behavior it follows naturally the interpretation of the entropy of the predictive class distribution as a measure for uncertainty quantification [37, 38].

The ECE metric considers only the confidence value assigned to the top-rated class. This value represents the class probability estimated by the classifier. While it is a confidence value, it does not represent the 'confidence' of the classifier on the prediction, it just concerns the predicted class in particular. Conversely, the entropy of the prediction vector is a measure of uncertainty of the whole prediction—i.e., an alternative and more comprehensive way of assessing the classifier's confidence in some prediction.

For instance, we may have two predictions $q^{(i)} = [0.6, 0.2, 0.2]$ and $q^{(j)} = [0.6, 0.4, 0.0]$ in a 3-class problem. Both assign the same confidence 0.6 to class 1, but it is clear that $q^{(i)}$ is a higher entropy predictive than $q^{(j)}$—i.e., it is a more uncertain prediction.

2.2 Post-hoc calibration

Ideally, a model f trained on some data $\mathcal{D}$ would *generalize* and show good calibration properties when evaluated on other data $\mathcal{D}_{\text{test}}$, assuming both sets are reasonably similar. However, many classification systems turn out to be badly calibrated in practice; for instance, Convolutional Neural Networks (CNNs) tend to produce overconfident predictions [6, 21]. Moreover, in some tasks, it cannot be guaranteed that the training data is similar enough to the actual data on which the model will be deployed. For instance, a language recognition system may be trained on broadcast narrowband speech (BNBS) data but applied in a telephone service where the audio characteristics are different. To solve this problem, one common approach is that of post-hoc calibration, in which a function is applied to the outputs of the model. This function can be seen as a decoupled classifier that learns to map uncalibrated outputs to calibrated ones—i.e., $q \mapsto \hat{q}$. We use the $\hat{}$ notation to denote the *calibrated* prediction. The standard practice is to fit this calibration map, or simply *calibrator*, in a held-out data set $\mathcal{D}_{cal}$, also called calibration data, that is supposed to resemble the data on which the model will make predictions.

Many post-hoc calibration methods take as input prediction *logits* instead of the final probability vectors. Notice that this does not limit their applicability since the outputs q of a probabilistic model can be mapped to the logit domain through the logarithmic function $z = \log q + k$, where k is an arbitrary scalar value and the logarithm is applied element-wise. Since post-hoc calibration only requires access to the outputs of the uncalibrated model it is common practice to work directly with logit-groundtruth pairs (z, y) directly. So, throughout this work when we talk about the calibration set we might refer to the set defined as:

$$\mathcal{D}'_{\text{cal}} = \{(f(x), y) | (x, y) \in \mathcal{D}_{\text{cal}}\},$$

where $z = f(x)$ are the logit outputs of the uncalibrated model. For the sake of simplicity we will refer to this as just $\mathcal{D}_{\text{cal}}$.

The standard procedure for post-hoc calibration is layed out in Algorithm 1. Note that we can use the same trainClassifier function to train the classification model and the calibration model. This is because, as we explained above, post-hoc calibration can be seen as a classification task in which the inputs are the outputs of an uncalibrated model.

Algorithm 1 Procedure to train and calibrate a classifier.

```
Require: $f$                                    ▷ Classification model to calibrate
Require: $\mathcal{D}$                                              ▷ Dataset
Require: $c$                                            ▷ Calibration model
  $\mathcal{D}_{train}, \mathcal{D}_{cal} \leftarrow \text{split}(\mathcal{D})$        ▷ Split dataset into 2 disjoint sets
  $f \leftarrow \text{trainClassifier}(f, \mathcal{D}_{train})$
  $\mathcal{D}_{cal} \leftarrow f(\mathcal{D}_{cal})$      ▷ Precompute uncalibrated outputs for calibration
  $c \leftarrow \text{trainClassifier}(c, \mathcal{D}_{cal})$
```

2.2.1 Accuracy-preserving calibration

Modern classification systems achieve very low test error rates and their miscalibration is attributed mainly to overconfidence—i.e., predicted confidences that call for higher accuracy rates than those actually obtained. Under this assumption, it is reasonable to constrain the calibration transforms so that the predicted ranking over the classes is maintained. This condition is known as accuracy-preserving [22] because functions that meet it do not change the top-label prediction:

$$\text{argmax}_k\, q_k = \text{argmax}_k\, \hat{q_k}.$$

When using expressive, and unconstrained, classification models like DNNs for the task of calibration, it is possible to improve calibration at the cost of losing accuracy [19, 23]. This trade-off is avoided by restricting the calibration functions to be accuracy-preserving so that the class decision, left to the classifier, is decoupled from the confidence estimation of each decision.

In this work, we compare only accuracy-preserving methods and avoid altogether the consequently trade-off often encountered in the calibration task: The question of determining which calibrator is better, one that improves more calibration but degrades the accuracy, or one that does not degrade the accuracy but shows less improvement on calibration. This decision is often application dependent but can be circumvented by using an accuracy-preserving method.

2.2.2 Temperature scaling

Temperature Scaling (TS) is probably the most widely used post-hoc calibration approach in the literature. It belongs to the family of accuracy-preserving methods. It scales the output logits by a *temperature factor* $T_0 \in \mathbb{R}^+$:

$$\hat{z} = \frac{z}{T_0}. \tag{6}$$

This factor is obtained by minimizing the NLL on some calibration data consisting of predictions of the uncalibrated classifier. Since the NLL is a Proper Scoring Rule, TS is encouraged to improve calibration. The simplicity of the method presents several advantages: The resulting optimization problem of its training has a closed solution and is cheap to compute. Moreover, it is an interpretable method by design since its temperature factor conveys information about the level of over-confidence of the classifier. A high temperature $T_0 > 1$ flattens the logits so the probability vectors approach the uniform distribution $q = [1/K, 1/K, ..., 1/K]$, thus relaxing the confidences and fixing over-confidence. On the other hand, a low temperature $T_0 < 1$ sharpens the confidence values moving the top-label predictions toward 1 and the others toward 0. Hence, fixing under-confidence.

3 Methods

In this section, we first describe the Adaptive Temperature Scaling family and illustrate it by proposing some methods of our contribution. Then, we introduce other accuracy-preserving methods, not necessarily of the ATS family, with state-of-the-art performance that we use as benchmarks in the experiments.

3.1 The adaptive temperature scaling family

We use the term ATS family to denote the group of accuracy-preserving maps that generalizes Temperature Scaling and can be expressed as:

$$\hat{z} = \frac{z}{T(z)}, \tag{7}$$

where $T : \mathbb{R}^K \mapsto \mathbb{R}^+$ is the *temperature function*.

This family generalizes Temperature Scaling by making the temperature factor input-dependent. TS is limited to the temperature function $T(z) = T_0$, where T_0 is the scalar parameter of the model. Hence, TS implicitly assumes that a classifier will generate predictions with the same level of over-confidence independently of the specific sample being classified.

On the other hand, a general ATS method computes a different temperature factor for each prediction via the temperature function $T(z)$. The computed factor for some z estimates the degree of over-confidence of the corresponding prediction $q = \sigma_{\text{SM}}(z)$. Hence, ATS methods acknowledge the possibility that a classifier's over-confidence may depend on the samples being classified.

The input-dependent property was first exploited by [29] with their Local Temperature Scaling method. However, this approach relies on the classifier input x to estimate a temperature factor $T_x = T(x)$. An ATS method estimates the factor based on the classifier output instead, $T_x = T(z)$, thus separating further the calibration step from the original classification task. The former approach tries to learn for which inputs x, e.g., images, the classifier is likely over-confident. ATS, on the other hand, is independent of the classification task requiring only access to precomputed predictions z. In other words, Local TS should be tailored for each classification task. For instance, if the input is audio one might use an RNN but choose a CNN instead for images. But both of them will output a logit vector in a classification task; thus, the input space of ATS methods is always the logit domain so these are more likely to generalize across classification tasks.

We acknowledge that this may reduce the potential expressive power of ATS since z is a processed version of x. Nevertheless, we believe that such constraint is not necessarily detrimental since, as we show in our experiments, the logit vector of a prediction already conveys information about its degree of miscalibration. Moreover, one advantage of post-hoc methods is the decoupling of the classification step from the calibration step. This is in some sense lost if the original classifier input is required for the calibration.

3.2 Proposed methods

We introduce three different ATS methods based on simple temperature functions. For each method we provide an hypothesis of why it can be used for recalibration and how the corresponding temperature function could be interpreted. Before delving into the methods we note that to meet the positivity constraint on the temperature factor we apply the softplus function to the outputs of ATS calibration models:

$$\sigma_{SP}(a) = \ln(1 + e^a). \tag{8}$$

3.2.1 Linear temperature scaling

We call this method Linear Temperature Scaling (LTS) since it is based on a linear combination of the logit vector, its temperature function is given by:

$$T_{\mathrm{LTS}}(z) = \sigma_{SP}(w^L z + b), \tag{9}$$

where $w^L \in \mathbb{R}^K$ and $b \in \mathbb{R}$ are the learnable parameters of the model.

The weight vector w^L takes into account the score assigned to each class to determine the level of over-confidence. Hence, LTS can predict higher temperature factors for certain predicted classes than for others. The scalar parameter b allows LTS to recover the base TS by zeroing the w^L parameter.

We motivate this method by giving the following example: an uncalibrated classifier can make over-confidence predictions for only certain classes. Since LTS weights each component of the logit vector to obtain the temperature factor, it should be able to raise (shrink) it by increasing (decreasing) the weight component w^L_i depending on whether the classifier is more (less) likely to make an over-confident prediction when predicting class i.

From this follows the interpretation of the method. After fitting LTS on a calibration set, the weight vector will point toward the direction of the highest degree of overconfidence in the logit space.

3.2.2 Entropy-based temperature scaling

Motivated by the fact that the entropy of the predictive class distribution can be interpreted as the uncertainty of such prediction, we propose HTS. The temperature function of this method is given by:

$$T_{\mathrm{HTS}}(z) = \sigma_{SP}\left(w^H \log \overline{H}(z) + b\right), \tag{10}$$

where $\overline{H}(z) = H(\sigma_{\mathrm{SM}}(z))/\log K$ is the *normalized entropy*, and $w^H \in \mathbb{R}$ and $b \in \mathbb{R}$ are the learnable parameters of the model. We normalize the entropy so that it is always upper-bounded by 1 irrespective of the number of classes. This allows us to generalize the interpretation of w^H between tasks with a different number of classes. We apply the logarithm to the entropy because, as we show later in the experiments, the temperature shows a linear trend with the logarithm of the entropy. We give b the same interpretation as in the previous model. The parameter w^H determines how much the uncertainty of the prediction vectors—i.e., the $\log \overline{H}(z)$—influences the temperature factor. The higher the magnitude of w^H the more variability we can expect in the computed temperature factors. On the other hand, a model with $w^H \to 0$ will resemble the base TS.

We motivate the method with a hypothetical example. Suppose that we have a classifier that produces predictions with variable degrees of over-confidence. One way in which a prediction-logit can convey information about its level of over-confidence is via its entropy. This is, for two predictions with the same predicted confidence, we may assume that the more uncertain of the 2—i.e., the higher entropy prediction—is more likely to be over-confident since it reports the same value of confidence despite its higher uncertainty.

This model makes a strong assumption about the level of over-confidence in a prediction. Mainly, that it can be expressed as a simple linear function of the log-entropy. The resulting model is easy to train since the set of possible calibration functions, or hypothesis space, is constrained by the number of trainable parameters. However, its performance is completely conditioned on the assumption being met. We provide experiments validating the model and its assumptions in Sect. 4.

3.2.3 Combined system

Finally we propose HnLTS, a model that combines the previous two with a single temperature function given by:

$$T_{HnLTS}(z) = \sigma_{SP}\left(w^L z + w^H \log \overline{H}(z) + b\right), \tag{11}$$

where $w^L \in \mathbb{R}^K$, $w^H \in \mathbb{R}$, and $b \in \mathbb{R}$ are the learnable parameters to which we give the same interpretation as above.

The motivation behind this model is to increase the expressiveness of the system in a controlled way to see how this affects its performance and training procedure compared to the more simpler methods. The hypothesis space of this method is a combination of LTS and HTS so it should be able to, at least, recover the solution of either one and achieve the same or better performance. However, we argue that the increased hypothesis space also makes the model more difficult to train with higher data requirements and this may impact its applicability depending on the task at hand, specially in data-scarce scenarios.

3.3 Baseline methods

We now describe other accuracy-preserving methods already existing in the literature with state-of-the-art performance. Some of these, but not all of them, belong to the ATS family as they can be expressed in the general form given by Eq. 3.1.

3.3.1 Parameterized temperature scaling

parameterized Temperature Scaling (PTS) [15] is a specific instance of the ATS family in which the temperature function is conditioned to be a neural network (NN). The input to the NN is the logit vector sorted by decreasing value of confidence z^s. Sorting the logit vector makes the model order-invariant [23] simplifying the hypothesis space at the cost of losing the possibility of discriminating between classes—i.e., it cannot consider the predicted ranking over the classes. PTS can be expressed as an ATS method with temperature function:

$$T_{\mathrm{PTS}}(z) = NN(z^s), \tag{12}$$

where NN is the function defined by the neural network.

Instead of optimizing the parameters of the NN to minimize some PSR as other methods do, authors propose to minimize an ECE-based loss given by:

$$L_{\mathrm{ECE}} = \sum_{i=1}^{M} \frac{|B_i|}{|\mathcal{D}_{\mathrm{test}}|} \|\mathrm{acc}(B_i) - \mathrm{cof}(B_i)\|_2, \tag{13}$$

where B_i, $\mathrm{cof}(B_i)$, and $\mathrm{acc}(B_i)$, are defined as in Eq. 2.1.1. During training, samples are re-partitioned into B_i at each loss evaluation since the confidence is re-scaled differently.

In their experiments, authors always use the same architecture, a Multi-Layer Perceptron (MLP) with two 5-unit hidden layers. Authors limit the input size of the network to the 10 highest confidence values whenever the number of classes is greater than 10. We use the same architecture in our experiments.

3.3.2 Bin-wise temperature scaling

Bin-Wise Temperature Scaling (BTS) [39] is a histogram-based method that applies a different temperature factor to each bin of the histogram. First, test samples are partitioned into N bins according to their top-label confidence. Authors force a high-confidence bin that ranges from 0.999 to 1. The samples with predicted confidence below 0.999 are partitioned into the other $N-1$ intervals such that each bin contains the same number of samples.

This method can also be included in the ATS family. The temperature function in this case is just a look-up table that assigns the corresponding temperature factor to the input confidence value.

3.3.3 Ensemble temperature scaling

Ensemble Temperature Scaling (ETS) [22] obtains a new logit vector as a convex combination of the uncalibrated vector, a maximum entropy logit vector, and the temperature-scaled vector:

$$\hat{z} = w_1 \frac{z}{T_{\mathrm{ETS}}} + w_2 z + w_3 \frac{1}{K}, \quad \text{subject to } w_1 + w_2 + w_3 = 1; w_i \geq 0 \tag{14}$$

where w_1, w_2, w_3 are the learnable weights of the convex combination and T_{ETS} is the temperature parameter of the TS component. All the parameters are optimized en bloc to minimize some PSR.

This method is also an extension of the standard TS, however, it does not belong to the ATS family. This can be easily verified by noting that ATS methods compute for some logit vector a single scalar temperature factor which applies equally to every entry of the logit vector. On the other hand, ETS scales by a different temperature factor each component of the logit vector.

4 Experiments

We present two sets of experiments. First, we report a study of the proposed methods that motivate their design and present ways in which calibration performance improves with model complexity. With these, we give evidence that the logit vector conveys information about its degree of over-confidence and motivate the design of new calibration methods that take this into account. Then, we compare our methods with other state-of-the-art accuracy-preserving calibration techniques in different dataset-size settings to assess their robustness to data scarcity.

4.1 Setup

4.1.1 Datasets and tasks

We refer to *model-dataset* pairs as calibration tasks. So a task is composed of the predictions of a *model*, for instance a ResNet-101 [40], on a specific *dataset*, like CIFAR-100 [41]. Every dataset is partitioned into three splits: train, calibration, and test. The model of each task is trained using the train set and then it is used to generate predictions on the calibration and test sets. We evaluate a calibration method on a certain task using the following procedure: First we fit the calibration method using the predictions on the calibration set. Then we apply it to the test set predictions and compute metrics over these.

4.1.2 Training details

We use NLL as the optimization objective to fit calibrators. Additionally, in all tasks, we fit a second version of the PTS method, minimizing the ECE-based loss instead (see Sect. 3.3). All methods except TS and ETS are implemented in Pytorch [17] and optimized using Stochastic

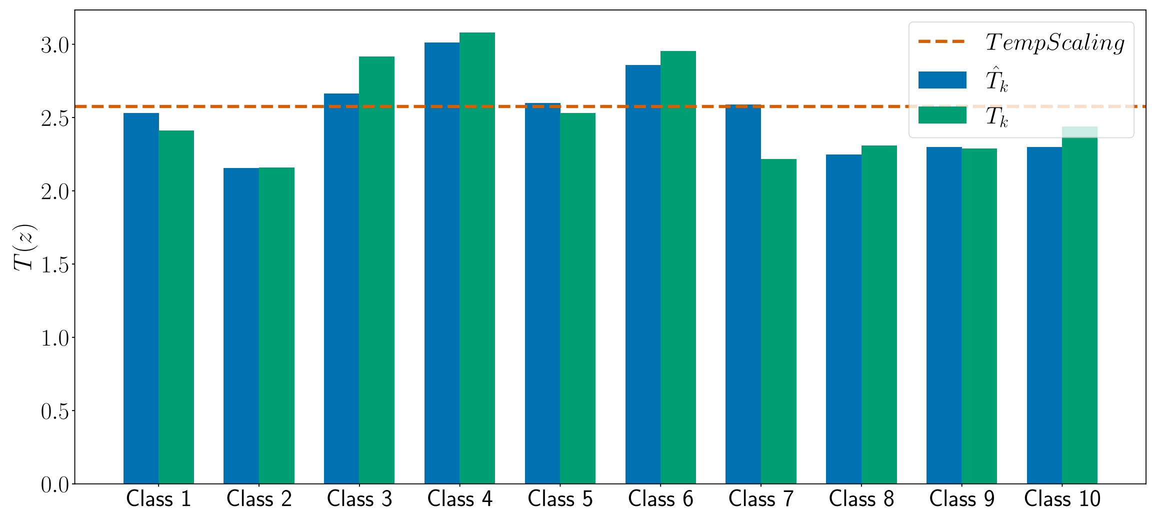


Fig. 2 Mean predicted temperature (blue) against optimum temperature factor (green) for each class on the test set

Gradient Descent (SGD) with an initial learning rate of 10^{-4}, Nesterov momentum [42] of 0.9, and a batch size of 1000. We reduce the learning rate on plateau by a factor of 10 until the learning rate reaches 10^{-7}, at that point we consider that the algorithm has converged and we stop training. These are default hyperparameters suggested by authors and provide good convergence across all calibration tasks in our experiments. The standard TS is optimized with SciPy [43] and the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm. To calibrate with ETS we use the code uploaded by authors [22].

4.2 Analysis of the ATS methods

For the first round of experiments, we first calibrate a ResNet-50 [40] on CIFAR-10 [41] with the proposed interpretable ATS methods, Entropy-based TS (HTS) and Linear TS (LTS), and discuss each separately. Then, we present an study of the relation between the number of parameters of a model and its data requirements for training. We calibrate a DenseNet-121 [44] trained on CIFAR-100 [41] with LTS, HTS, and HnLTS using varying number of calibration samples.

4.2.1 Linear TS: Introducing class dependence

With this experiment, we aim to illustrate the example that we give to motivate the LTS method. This is, that LTS can adapt to a classifier that makes more or less over-confident predictions depending on which class it predicts as correct.

First, we train a ResNet-05 [40] on the CIFAR-10 train dataset. Then, we use the calibration set to calibrate the model using TS and the LTS method.

To check our hypothesis, we divide the test set according to their true class:

$$\mathcal{D}_k = \{(x, y) \in \mathcal{D}_{\text{test}} | \text{argmax}(y) = k\},$$

and compute for each subset $\mathcal{D}_k$ the optimum temperature factor T_k, which is obtained by optimizing TS directly on $\mathcal{D}_k$.

Then, we use the LTS model optimized on the calibration set to compute a temperature factor for every test prediction, group them according to their true class, and take the average for every group:

$$\hat{T}_k = \frac{1}{|\mathcal{D}_k|} \sum_{(x \in \mathcal{D}_k)} (T_{\text{LTS}}(f(x))).$$

Finally, we represent in Fig. 2 $\hat{T}_k$ against the optimum temperature T_k. For reference, we include the TS temperature factor learned on the calibration set (dashed orange line).

From Fig. 2 we notice that the classifier does produce more over-confident predictions for some classes than for others, even in a curated and well-balanced dataset such as CIFAR-10. We can expect this effect to be even more present in real-life applications in which the prevalence of classes may vary and some distribution mismatch between development and production data can be expected. LTS exploits this difference between classes and manages to adapt the temperature factor in each subset closely matching the optimum.

4.2.2 Entropy-based TS: Leveraging uncertainty of predictions

Our motivation for the HTS method is that the level of over-confidence in a prediction is related to the entropy of such prediction. If our hypothesis is correct, we can expect, for the same value of confidence in the predicted class, higher entropy predictions to be more over-confident on

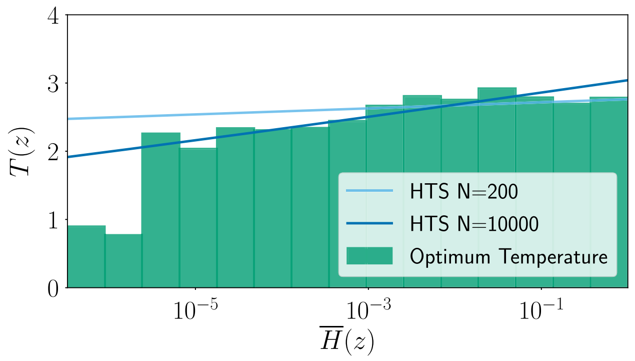


Fig. 3 Temperature function of HTS fitted using 200 (light blue) and 10000 (dark blue) calibration samples and optimal temperature on the test set (green)

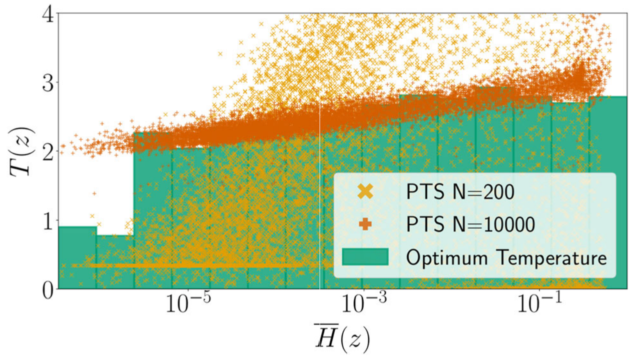


Fig. 4 Temperature factors of PTS for test samples fitted using 200 (light orange) and 10000 (dark orange) calibration samples and optimal temperature on the test set (green)

average. So, we might expect higher temperature factors for higher entropy predictions. The relation between overconfidence and entropy has already been explored and exploited in [38]. However, in this case, authors penalize low-entropy predictions as over-confident. Our hypothesis does not necessarily oppose to theirs; we suggest that the inverse relation exists for fixed confidence values, while theirs account for a general relationship across confidence values. The relation is driven in this case by the fact that high values of confidence for a given class imply low entropy in the prediction.

In Fig. 3 we depict the temperature function learned by HTS in the calibration set. We train two models, one with the full calibration set, plotted in a darker shade, and the other using a random subset of 200 samples. We also plot the optimum temperature factor estimated in the test set for different ranges of normalized entropy. We partition the log-domain of the normalized entropy into 15 equally spaced bins and divide the test samples according to this binning scheme $\mathcal{D}_{\text{test}} = \{\mathcal{D}_1, \ldots, \mathcal{D}_{15}\}$. For each set $\mathcal{D}_i$, we estimate the optimum temperature T_i factor given by TS in the same way we do in the first experiment:

$$T_i = TS(\mathcal{D}_i).$$

Note how for values of $\log \overline{H} \geq 10^{-5}$ the optimum temperature increases linearly with the log-entropy, and that this behavior is captured by the *temperature function* learnt by HTS. We also notice that when there is plenty of calibration data, the model captures better the linear trend, but when data is limited the slope flattens—i.e., $w^H/to0$—approaching the standard TS.

We also fit the PTS method on the same subsets as HTS resulting in 2 calibration methods, PTS trained with 200 samples and PTS trained with the full 10000 calibration samples. Then we plot the temperature factor that each PTS assigns to each prediction on the test set on Fig. 4. With this plot, we aim to see if a very expressive method like PTS learns any relation between the entropy of a prediction and its temperature factor. Note how the model trained with full data produce temperature factors that resemble the linear trend showed by HTS, but that when data is scarce PTS fails to grab any relation and produce very variable factors.

We find that, at least in this particular task, there exists some positive relation between the entropy of the predictive and its level of over-confidence. Figure 3 shows that a linear function is a fair approximation to the relation between entropy and temperature and that HTS manages to capture it even in the face of low data. We can check in Table 1 that PTS and HTS are highly correlated when both methods are trained using the whole calibration set. Moreover, if HTS is trained using only a small subset of the data ($N = 200$), this method still produces temperature factors highly correlated with those issued by PTS on a large set. On the other hand, the temperature factors given by PTS trained with low-data ($N = 200$) are much less correlated with those given by the full-data trained PTS. This suggests that the behavior that PTS has to learn from abundant data, is already ingrained in the inductive bias of HTS.

In Fig. 4 we show that a much more expressive method like PTS also captures this linear relationship when given enough data. However, in the face of limited data, it fails to do so. Moreover, in Fig. 5 we plot for all samples in the test set the temperature factors given by PTS against those by HTS, both methods fitted using all samples. The plot shows that when data is plentiful the function learnt by both methods is reasonable similar, suggesting that the function space of HTS contains well-performing solutions similar to those learnt by PTS despite being much more constrained.

Table 1 Sample Pearson correlation coefficient between the temperatures given by HTS and PTS temperature functions on the test set.

r_{xy}	HTS (N=1000)	HTS (N=200)	PTS (N=200)
PTS (N=1000)	0.89025	0.89060	0.19945

All coefficients are computed against the temperatures given by the PTS method trained with the full set (N=10000)

4.2.3 Performance degradation with model complexity

In order to validate our hypothesis that more complex methods require more data and are inherently more difficult to fit, we conduct the following experiment. We use a DenseNet-121 [44] trained on CIFAR-100 train dataset to generate predictions on a calibration set, disjoint from the training set, of 10000 samples. From it we sample 2 additional subsets with 200 and 1000 samples respectively, having in total 3 calibration sets with increasing number of samples. We use each calibration set to fit the TS, LTS, HTS and HnLTS methods giving a total of 12 different combinations.

We compare LTS, a highly parameterized model, hence more expressive, with HTS, a 2 parameter model but with a strong inductive bias. We also include HnLTS that acts as a control in the following sense: If there is a difference in performance between LTS and HTS in low-data scenarios, one could argue that LTS is not expressive enough and that its complexity does not hinder the training. Would that be the case, HnLTS should be able to match the HTS performance since its function space includes all solutions of both models, HTS and LTS. So we expect HnLTS to fail in the same low-data scenerarios as LTS because of being highly parameterized. The TS scaling method is included as baseline.

We evaluate all of them in the CIFAR-100 test dataset and report performance in terms of NLL and ECE using ($M = 50$) bins in Tables 2 and 3 respectively for each scenario we mark the score (either NLL or ECE) of the best performing model in bold.

We first note that when data is plentiful ($N = 10000$) all calibration models achieve similar performance both in terms of NLL and ECE. But as we reduce the number of calibration samples ($N = 1000$ and $N = 200$), the performance of both LTS and HnLTS are greatly impacted. On the other hand, its remarkable how with only $N = 200$ calibration samples both simple models maintain a similar performance.

4.3 Benchmarking

In this section, we compare the performance of the proposed ATS methods: LTS, HTS, and HnLTs; with state-of-the-art accuracy-preserving methods: TS, ETS, BTS, and PTS. We fit two versions of PTS: One trained to minimize the NLL, the calibration objective we use to train every method; and a second version optimizing the ECE-based loss instead as reported in [15](see Sect. 3.3). We refer to the former as PTS and the latter as PTSe where the 'e' stands for the ECE-based objective.

All experiments are run 50 times with different random initializations and the results are averaged across runs. For experiments in which we subsample the calibration set to simulate data-scarcity scenarios, we sample one subset at each run and use that subset to train all calibration

```
Require: $f$                                        ▷ Classification model to calibrate
Require: $\mathcal{D}$                                        ▷ Samples from dataset
Require: $\mathcal{C}$                               ▷ Set of calibration models to evaluate
Require: $N$                                 ▷ Number of calibration samples to use
  $\mathcal{D}_{train}, \mathcal{D}_{cal}, \mathcal{D}_{test} \leftarrow \text{split}(\mathcal{D})$           ▷ Split dataset into 3 disjoint sets
  $f \leftarrow \text{trainClassifier}(f, \mathcal{D}_{train})$
  $\mathcal{D}_{cal} \leftarrow f(\mathcal{D}_{cal})$          ▷ Precompute uncalibrated outputs for calibration
  $\mathcal{D}_{test} \leftarrow f(\mathcal{D}_{test})$           ▷ Precompute uncalibrated outputs for testing
  for $i = 1$ to $i = 50$ do
    $\mathcal{D}' \leftarrow \text{subsample}(\mathcal{D}_{cal}, N)$
    for $c$ in $\mathcal{C}$ do
      $c \leftarrow \text{trainClassifier}(c, \mathcal{D}')$
      $\text{metrics}[c][i] \leftarrow \text{evaluate}(c, \mathcal{D}_{test})$
    end for
  end for
  for $c$ in $\mathcal{C}$ do
    $\text{metrics}[c] \leftarrow \text{average}(\text{metrics}[c])$
  end for
```

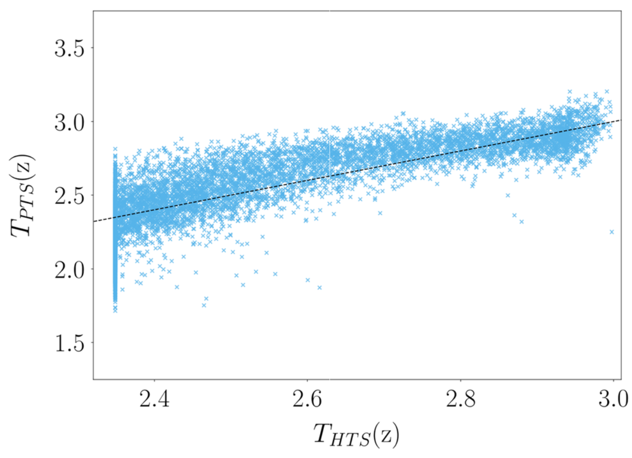


Fig. 5 Temperature factor computed by PTS against temperature factor computed by HTS for test set predictions. The black dotted line represents a perfect one to one relation

Table 2 NLL for different calibration models

Calibration samples	TS	LTS	HTS	HnLTS
$N = 200$	0.837	–	**0.829**	–
$N = 1000$	0.836	0.922	**0.826**	0.899
$N = 10000$	0.835	0.836	0.825	**0.819**

Table 3 ECE (M=50) for different calibration models

Calibration samples	TS	LTS	HTS	HnLTS
$N = 200$	4.01	33.77	**3.56**	25.95
$N = 1000$	3.85	5.67	**3.34**	5.03
$N = 10000$	3.93	4.20	**3.31**	3.48

methods. So, in each of the 50 runs, all the calibration methods see the same N-sized calibration set, but this subset is different across tasks. The training procedure is layout in Algorithm 2.

Algorithm 2 Procedure to evaluate calibration models on a model-dataset task

4.3.1 Results

For the sake of space and simplicity, we depict results for each dataset and average across models—e.g., average ECE of HTS on all CIFAR-10 tasks. We defer detailed results to Appendix 1. Results are shown in Fig. 6. We normalize each metric by the performance of TS as we consider it the main benchmark. The performance of the base TS is computed in the same conditions as the evaluated calibrator—i.e., trained and evaluated using the same calibration and test sets. We report performance in terms of normalized ECE and normalized NLL, namely $\overline{ECE}$ and $\overline{NLL}$. For each method, we plot five markers, the size of which increases with the size of the data set. From smallest to biggest these are $N = (200, 500, 1000, 5000, 10000)$. The y-axis position of the marker indicates the mean value across tasks, where each task is a different NN architecture calibrated.

We first point out that almost all models outperform the simple TS when there is enough data (big markers), although, on average, there are no big differences between models. However, when data is scarce all the highly-parameterized models show severe performance degradation and only ETS and HTS seem to provide consistent performance. Moreover HTS provides better results in most of the individual tasks while ETS barely outperforms the baseline TS.

Also, it is worth noting the difference between datasets. In the highly dimensional CIFAR100, we can see a greater advantage in using calibration methods more complex than TS. On the other hand, the best methods barely outperform TS in CIFAR10 tasks. This suggests that the problem of calibration may grow more complex with the number of classes, although the number of datasets included in our experiments is limited and more experiments are required to validate this observation.

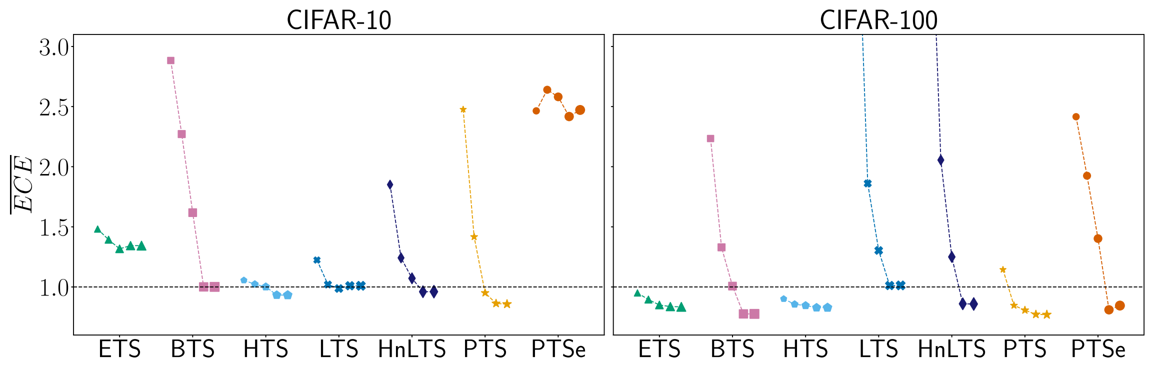
Interestingly, HnLTS fails in low-data scenarios, even though it could, in theory, recover the HTS solution by zeroing the w^L parameter. This suggests that increasing expressiveness can do more harm than good by complicating the training objective.

5 Conclusions

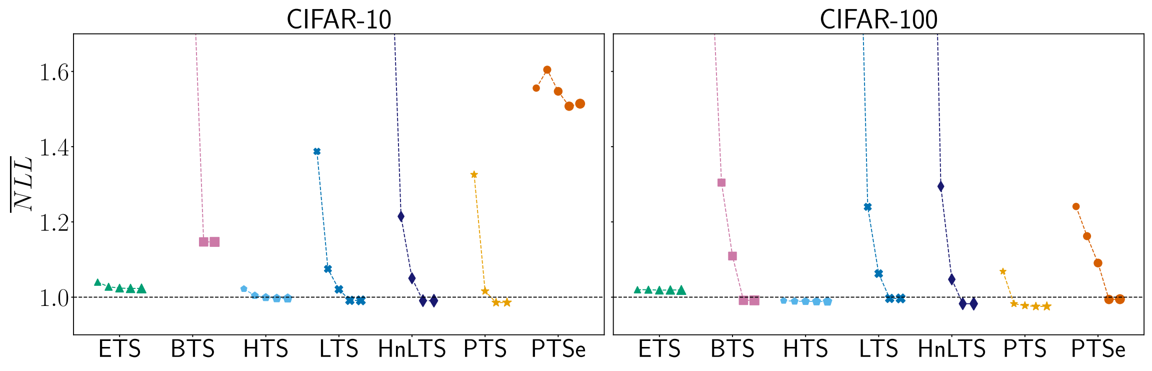
We have shown that post-hoc calibration of DNNs can benefit from more expressive models than the widely used Temperature Scaling, especially in tasks with a high number of classes. For instance, simply adjusting the temperature factor of TS with a linear combination of the logit prediction improves calibration by taking into account the score assigned to each class.

However, more complex models require higher amounts of data to find a good-performing solution. This poses a trade-off between the complexity of the calibration model and the available data to train the model. There are many real-world tasks where data for re-calibration is limited and hinders the calibration with a complex model.

By analyzing the calibration functions learned by expressive models on plenty of data, we can design simpler



(a) Average Relative ECE in CIFAR tasks.



(b) Average Relative NLL in CIFAR tasks.

Fig. 6 Average results for CIFAR-10 (left) and CIFAR-100 (right) tasks of all calibration methods in terms of ECE (top) and NLL (bottom) normalized by the performance of TS, namely $\overline{ECE}$ and $\overline{NLL}$

models with a strong inductive bias toward similar calibration functions. In this work, we have introduced HTS, a 2-parameter model that scales predictions according to their entropy. The temperature factors estimated by PTS, a much more expressive model, follow the same linear relation with the predictive entropy that HTS implicitly assumes. HTS shows calibration performance comparable to that of more expressive methods on ideal data conditions. However, the application of these other methods is limited to conditions where data is plentiful and one has to resort to simpler methods like TS when data is limited. On the other hand, HTS maintains good performance across all ranges of data availability making it a good default for calibration. Moreover, an important feature of the model is that it is interpretable, characterizing the link between a prediction's uncertainty and its over-confidence.

With this work, we motivate the study of expressive methods as a way to design practical models with a suitable inductive bias. As a first approach, we propose to use a hand-designed low-parameter model to achieve this bias. This model achieves comparable performance to other state-of-the-art methods while being robust to low data scenarios. In future work, we plan to try other forms of inducing the desired bias, for instance, via the prior specification in a Bayesian inference setting. This option may allow training higher capacity models while still being robust to data scarcity.

Appendix

Results

In this section, we provide in tables the results for each model-dataset task. Additionally, we give average performance normalized by that of the uncalibrated model across tasks in each dataset.

Table 4 ECE ($M = 50$)

	Task	Uncalibrated	TS	ETS	BTS	HTS	LTS	HnLTS	PTS	PTS (L_{ECE})
CIFAR 10	DenseNet 121	2.84	1.39	2.26	**1.09**	1.41	1.30	1.35	1.13	3.20
	DenseNet 169	2.79	1.65	2.31	1.27	1.10	1.29	1.02	**0.91**	2.18
	ResNet 50	10.71	2.54	2.84	1.86	**1.62**	2.43	2.12	1.66	6.61
	ResNet 101	4.45	**1.37**	1.49	1.72	1.54	1.42	1.47	1.43	4.20
	ResNext 29 8x16	2.88	1.03	**0.85**	1.19	1.06	1.11	1.15	1.01	3.39
	VGG 19	4.61	2.51	3.30	**1.38**	1.62	2.15	1.67	1.49	5.02
	WRN 28x10	1.93	**0.70**	1.59	1.07	0.88	0.88	0.87	0.86	1.25
	WRN 40x10	3.12	1.04	1.14	1.28	1.11	1.21	1.21	**1.03**	3.55
	Avg. Relative $\overline{ECE}$	1.00	0.40	0.55	0.38	0.37	0.40	0.37	**0.33**	0.94
CIFAR 100	DenseNet 121	8.76	3.93	2.96	**2.77**	3.27	4.12	3.23	3.22	3.26
	DenseNet 169	8.93	3.95	2.85	3.19	3.36	4.15	3.67	3.21	**2.76**
	ResNet 101	11.45	2.25	2.22	2.22	2.32	2.66	2.32	**1.92**	2.44
	ResNext 29 8x16	9.69	3.14	2.80	2.06	2.09	3.51	2.55	**1.79**	2.79
	VGG 19	17.63	5.13	5.36	3.89	3.78	3.60	3.56	**3.28**	4.22
	WRN 28x10	5.19	4.63	3.64	**3.11**	3.52	4.59	3.69	3.33	4.10
	WRN 40x10	14.78	4.20	**2.76**	3.55	3.85	4.19	3.87	4.09	2.95
	Avg. Relative $\overline{ECE}$	1.00	0.41	0.33	**0.31**	0.33	0.41	0.35	0.31	0.34

Models are denoted by their architecture and depth (and width if applicable)

Bold would be the best performing model

Table 5 NLL

	Task	Uncalibrated	TS	ETS	BTS	HTS	LTS	HnLTS	PTS	PTS (L_{ECE})
CIFAR 10	DenseNet 121	0.1881	0.1618	0.1673	0.1835	0.1611	0.1606	0.1604	**0.1585**	0.2407
	DenseNet 169	0.1870	0.1608	0.1686	0.2035	0.1590	0.1593	0.1584	**0.1542**	0.1934
	ResNet 50	0.7897	0.4473	0.4515	0.4498	0.4447	0.4399	**0.4385**	0.4444	0.6847
	ResNet 101	0.3047	0.2163	0.2199	0.2411	0.2173	**0.2132**	0.2142	0.2168	0.3570
	ResNext 29 8x16	0.1997	0.1621	**0.1617**	0.1909	0.1623	0.1635	0.1634	0.1628	0.2820
	VGG 19	0.2998	0.2355	0.2424	0.2672	0.2330	0.2313	0.2299	**0.2296**	0.4191
	WRN 28x10	0.1497	0.1362	0.1424	0.1750	0.1364	0.1348	0.1355	**0.1346**	0.1390
	WRN 40x10	0.2068	0.1634	0.1642	0.1729	0.1634	0.1632	0.1633	**0.1607**	0.2783
	Avg. Relative $\overline{NLL}$	1.0000	0.7876	0.8060	0.9096	0.7842	0.7804	0.7798	**0.7750**	1.1797
CIFAR 10	DenseNet 121	0.8939	0.8355	0.8589	0.8271	0.8253	0.8351	0.8209	**0.8161**	0.8321
	DenseNet 169	0.8748	0.8156	0.8328	0.8082	0.8061	0.8152	0.8021	**0.7950**	0.8112
	ResNet 101	1.1343	1.0007	1.0088	1.0040	1.0002	1.0061	1.0035	**0.9971**	1.0037
	ResNext 29 8x16	0.9398	0.8220	0.8404	0.8190	0.8128	0.8282	0.8182	**0.8044**	0.8173
	VGG 19	1.5414	1.1997	1.2069	1.1926	1.1941	1.1825	1.1821	**1.1737**	1.1939
	WRN 28x10	0.8173	0.8135	0.8343	0.7876	0.7868	0.8040	0.7765	**0.7719**	0.8054
	WRN 40x10	1.2248	0.9055	0.9261	0.9037	0.8987	0.8969	**0.8821**	0.8832	0.8959
	Avg. Relative $\overline{NLL}$	1.0000	0.8767	0.8938	0.8687	0.8661	0.8739	0.8611	**0.8548**	0.8720

Models are denoted by their architecture and depth (and width if applicable)

Bold would be the best performing model

Table 6 Brier Score

	Task	Uncalibrated	TS	ETS	BTS	HTS	LTS	HnLTS	PTS	PTS (L_{ECE})
CIFAR 10	DenseNet 121	0.0764	0.0729	0.0724	0.0722	0.0726	0.0727	0.0724	**0.0720**	0.0787
	DenseNet 169	0.0754	0.0717	0.0712	0.0711	0.0712	0.0714	0.0710	**0.0705**	0.0740
	ResNet 50	0.2392	0.2037	0.2029	0.2036	0.2027	0.2020	**0.2014**	0.2025	0.2233
	ResNet 101	0.1102	0.1011	0.1010	0.1019	0.1011	0.1004	**0.1004**	0.1007	0.1113
	ResNext 29 8x16	0.0828	0.0783	0.0782	0.0791	0.0784	0.0783	0.0783	**0.0781**	0.0869
	VGG 19	0.1101	0.1019	0.1007	0.1003	0.1005	0.1011	0.1000	**0.0999**	0.1139
	WRN 28x10	0.0629	0.0608	0.0609	0.0611	0.0607	0.0606	**0.0606**	0.0606	0.0612
	WRN 40x10	0.0820	0.0768	0.0764	0.0770	0.0767	0.0767	0.0767	**0.0764**	0.0853
	Avg. Relative $\overline{Brier}$	1.0000	0.9312	0.9275	0.9304	0.9275	0.9272	0.9245	**0.9231**	1.0069
CIFAR 100	DenseNet 121	0.3171	0.3048	0.3043	0.3052	0.3049	0.3040	0.3036	**0.3024**	0.3029
	DenseNet 169	0.3142	0.3017	0.3010	0.3020	0.3016	0.2999	0.2995	**0.2984**	0.2990
	ResNet 101	0.4053	0.3817	0.3816	0.3825	0.3817	0.3817	0.3814	**0.3805**	0.3817
	ResNext 29 8x16	0.3275	0.3096	0.3093	0.3096	0.3090	0.3100	0.3097	**0.3068**	0.3085
	VGG 19	0.4433	0.3918	0.3910	0.3880	0.3897	0.3882	0.3878	**0.3848**	0.3868
	WRN 28x10	0.2892	0.2886	0.2877	0.2871	0.2872	0.2851	0.2839	**0.2831**	0.2868
	WRN 40x10	0.3700	0.3276	0.3270	0.3293	0.3288	0.3240	0.3240	0.3245	**0.3239**
	Avg. Relative $\overline{Brier}$	1.0000	0.9394	0.9378	0.9387	0.9382	0.9341	0.9328	**0.9289**	0.9328

Models are denoted by their architecture and depth (and width if applicable)

Bold would be the best performing model

Table 7 ECE ($M = 50$) using 5000 samples

	Task	Uncalibrated	TS	ETS	BTS	HTS	LTS	HnLTS	PTS	PTS (L_{ECE})
CIFAR 10	DenseNet 121	2.84	1.39	2.26	**1.09**	1.41	1.29	1.34	1.10	2.99
	DenseNet 169	2.79	1.65	2.31	1.27	1.10	1.29	1.02	**0.97**	2.33
	ResNet 50	10.71	2.54	2.84	1.86	**1.62**	2.43	2.14	1.70	5.53
	ResNet 101	4.45	**1.37**	1.49	1.72	1.54	1.42	1.47	1.43	4.46
	ResNext 29 8x16	2.88	1.03	**0.85**	1.19	1.06	1.11	1.14	1.03	3.12
	VGG 19	4.61	2.51	3.30	**1.38**	1.62	2.15	1.67	1.41	5.08
	WRN 28x10	1.93	**0.70**	1.59	1.07	0.89	0.89	0.86	0.88	1.24
	WRN 40x10	3.12	1.04	1.14	1.28	1.11	1.21	1.22	**1.02**	3.64
	Avg. Relative $\overline{ECE}$	1.00	0.40	0.55	0.38	0.37	0.40	0.37	**0.33**	0.93
CIFAR 100	DenseNet 121	8.76	3.93	2.95	**2.77**	3.28	4.11	3.23	3.19	3.10
	DenseNet 169	8.93	3.95	**2.85**	3.19	3.36	4.16	3.67	3.20	2.91
	ResNet 101	11.45	2.25	2.23	2.22	2.30	2.65	2.34	**1.90**	2.41
	ResNext 29 8x16	9.69	3.14	2.83	2.06	2.09	3.50	2.56	**2.01**	2.45
	VGG 19	17.63	5.13	5.36	3.89	3.79	3.58	3.58	**3.28**	3.82
	WRN 28x10	5.19	4.63	3.65	**3.11**	3.54	4.60	3.68	3.23	4.08
	WRN 40x10	14.78	4.20	**2.77**	3.55	3.86	4.19	3.87	4.03	2.81
	Avg. Relative $\overline{ECE}$	1.00	0.41	0.33	**0.30**	0.33	0.41	0.35	0.31	0.33

Models are denoted by their architecture and depth (and width if applicable)

Bold would be the best performing model

Table 8 NLL using 5000 samples

	Task	Uncalibrated	TS	ETS	BTS	HTS	LTS	HnLTS	PTS	PTS (L_{ECE})
CIFAR 10	DenseNet 121	0.1881	0.1618	0.1673	0.1835	0.1611	0.1606	0.1604	**0.1582**	0.2267
	DenseNet 169	0.1870	0.1608	0.1686	0.2035	0.1590	0.1593	0.1584	**0.1545**	0.2111
	ResNet 50	0.7897	0.4473	0.4515	0.4498	0.4447	0.4399	**0.4385**	0.4446	0.7042
	ResNet 101	0.3047	0.2163	0.2199	0.2411	0.2173	**0.2132**	0.2142	0.2169	0.3689
	ResNext 29 8x16	0.1997	0.1621	**0.1617**	0.1909	0.1623	0.1635	0.1634	0.1627	0.2557
	VGG 19	0.2998	0.2355	0.2424	0.2672	0.2330	0.2313	0.2299	**0.2291**	0.4007
	WRN 28x10	0.1497	0.1362	0.1424	0.1750	0.1364	0.1348	0.1355	**0.1347**	0.1423
	WRN 40x10	0.2068	0.1634	0.1642	0.1729	0.1634	0.1632	0.1633	**0.1608**	0.2849
	Avg. Relative $\overline{NLL}$	1.0000	0.7867	0.8060	0.9096	0.7842	0.7804	0.7798	**0.7748**	1.1727
CIFAR 100	DenseNet 121	0.8939	0.8355	0.8587	0.8271	0.8253	0.8351	0.8209	**0.8152**	0.8331
	DenseNet 169	0.8748	0.8156	0.8328	0.8082	0.8061	0.8152	0.8021	**0.7943**	0.8104
	ResNet 101	1.1343	1.0007	1.0088	1.0040	1.0002	1.0060	1.0035	**0.9972**	1.0044
	ResNext 29 8x16	0.9398	0.8220	0.8404	0.8190	0.8128	0.8282	0.8182	**0.8065**	0.8144
	VGG 19	1.5414	1.1997	1.2069	1.1926	1.1941	1.1826	1.1821	**1.1751**	1.1903
	WRN 28x10	0.8173	0.8135	0.8342	0.7876	0.7868	0.8041	0.7765	**0.7691**	0.8069
	WRN 40x10	1.2248	0.9055	0.9261	0.9037	0.8987	0.8969	**0.8821**	0.8828	0.8943
	Avg. Relative $\overline{NLL}$	1.0000	0.8767	0.8937	0.8687	0.8661	0.8739	0.8611	**0.8545**	0.8714

Models are denoted by their architecture and depth (and width if applicable)

Bold would be the best performing model

Table 9 Brier Score using 5000 samples

	Task	Uncalibrated	TS	ETS	BTS	HTS	LTS	HnLTS	PTS	PTS (L_{ECE})
CIFAR 10	DenseNet 121	0.0764	0.0729	0.0724	0.0722	0.0726	0.0727	0.0724	**0.0720**	0.0778
	DenseNet 169	0.0754	0.0717	0.0712	0.0711	0.0712	0.0714	0.0710	**0.0706**	0.0747
	ResNet 50	0.2392	0.2037	0.2029	0.2036	0.2026	0.2020	**0.2014**	0.2026	0.2199
	ResNet 101	0.1102	0.1011	0.1010	0.1019	0.1011	0.1004	**0.1004**	0.1007	0.1123
	ResNext 29 8x16	0.0828	0.0783	0.0782	0.0791	0.0784	0.0783	0.0783	**0.0781**	0.0856
	VGG 19	0.1101	0.1019	0.1007	0.1003	0.1005	0.1011	0.1000	**0.0998**	0.1139
	WRN 28x10	0.0629	0.0608	0.0609	0.0611	0.0607	0.0606	**0.0606**	0.0606	0.0614
	WRN 40x10	0.0820	0.0768	0.0764	0.0770	0.0767	0.0767	0.0767	**0.0764**	0.0858
	Avg. Relative $\overline{Brier}$	1.0000	0.9312	0.9275	0.9304	0.9275	0.9272	0.9245	**0.9231**	1.0048
CIFAR 100	DenseNet 121	0.3171	0.3048	0.3043	0.3052	0.3049	0.3040	0.3036	**0.3023**	0.3029
	DenseNet 169	0.3142	0.3017	0.3010	0.3020	0.3016	0.2999	0.2995	**0.2982**	0.2991
	ResNet 101	0.4053	0.3817	0.3816	0.3825	0.3817	0.3817	0.3814	**0.3805**	0.3818
	ResNext 29 8x16	0.3275	0.3096	0.3093	0.3096	0.3090	0.3100	0.3097	**0.3071**	0.3079
	VGG 19	0.4433	0.3918	0.3910	0.3880	0.3897	0.3882	0.3878	0.3851	**0.3850**
	WRN 28x10	0.2892	0.2886	0.2877	0.2871	0.2872	0.2851	0.2839	**0.2827**	0.2871
	WRN 40x10	0.3700	0.3276	0.3270	0.3293	0.3288	0.3240	0.3240	0.3245	**0.3232**
	Avg. Relative $\overline{Brier}$	1.0000	0.9394	0.9378	0.9387	0.9382	0.9341	0.9328	**0.9289**	0.9320

Models are denoted by their architecture and depth (and width if applicable)

Bold would be the best performing model

Table 10 ECE ($M = 50$) using 1000 samples

	Task	Uncalibrated	TS	ETS	BTS	HTS	LTS	HnLTS	PTS	PTS (L_{ECE})
CIFAR 10	DenseNet 121	2.84	1.52	2.29	2.12	1.51	**1.30**	1.49	1.34	3.09
	DenseNet 169	2.79	1.55	2.40	1.79	1.32	**1.20**	1.35	1.22	3.08
	ResNet 50	10.71	2.56	2.75	3.19	2.16	2.38	2.22	**1.90**	7.79
	ResNet 101	4.45	**1.46**	1.57	2.72	1.66	1.64	1.73	1.53	4.09
	ResNext 29 8x16	2.88	**1.04**	1.08	2.24	1.28	1.29	1.40	1.25	3.16
	VGG 19	4.61	2.54	3.20	2.65	1.99	1.99	1.95	**1.72**	4.88
	WRN 28x10	1.93	**0.87**	1.57	1.68	0.91	0.88	1.14	0.97	2.24
	WRN 40x10	3.12	**1.16**	1.23	2.30	1.29	1.32	1.39	1.29	3.49
	Avg. Relative $\overline{ECE}$	1.00	0.42	0.57	0.66	0.42	0.40	0.44	**0.39**	1.04
CIFAR 100	DenseNet 121	8.76	3.85	**3.01**	3.78	3.20	5.25	4.76	3.21	5.67
	DenseNet 169	8.93	4.07	**2.97**	3.82	3.31	5.31	5.04	3.24	4.78
	ResNet 101	11.45	2.28	2.26	3.32	2.47	4.03	4.20	**2.09**	4.08
	ResNext 29 8x16	9.69	3.20	2.80	3.24	2.26	4.82	4.48	**2.06**	3.13
	VGG 19	17.63	5.08	5.25	4.20	4.02	4.00	4.61	**3.96**	6.14
	WRN 28x10	5.19	4.69	3.72	3.79	3.59	5.26	4.56	**3.38**	6.16
	WRN 40x10	14.78	4.34	**3.16**	4.46	4.09	5.46	5.00	4.21	7.89
	Avg. Relative $\overline{ECE}$	1.00	0.42	0.34	0.39	0.34	0.52	0.49	**0.33**	0.56

Models are denoted by their architecture and depth (and width if applicable)

Bold would be the best performing model

Table 11 NLL using 1000 samples

	Task	Uncalibrated	TS	ETS	BTS	HTS	LTS	HnLTS	PTS	PTS (L_{ECE})
CIFAR 10	DenseNet 121	0.1881	0.1621	0.1685	0.3142	**0.1618**	0.1651	0.1710	0.1648	0.2331
	DenseNet 169	0.1870	0.1611	0.1685	0.2973	0.1598	0.1649	0.1710	**0.1594**	0.2525
	ResNet 50	0.7897	0.4477	0.4520	0.5999	0.4461	0.4441	**0.4438**	0.4476	0.7613
	ResNet 101	0.3047	**0.2166**	0.2199	0.4069	0.2175	0.2188	0.2217	0.2200	0.3255
	ResNext 29 8x16	0.1997	**0.1624**	0.1633	0.3835	0.1635	0.1719	0.1775	0.1730	0.2420
	VGG 19	0.2998	0.2358	0.2421	0.4146	**0.2337**	0.2371	0.2426	0.2352	0.3991
	WRN 28x10	0.1497	**0.1364**	0.1419	0.2575	0.1367	0.1389	0.1491	0.1380	0.1730
	WRN 40x10	0.2068	**0.1637**	0.1653	0.3747	0.1642	0.1701	0.1730	0.1690	0.2813
	Avg. Relative $\overline{NLL}$	1.0000	0.7880	0.8080	1.5240	**0.7874**	0.8053	0.8304	0.8010	1.2101
CIFAR 100	DenseNet 121	0.8939	0.8359	0.8601	0.9227	0.8260	0.9021	0.8849	**0.8161**	0.9383
	DenseNet 169	0.8748	0.8161	0.8326	0.9240	0.8070	0.8797	0.8682	**0.7971**	0.8577
	ResNet 101	1.1343	1.0010	1.0084	1.0979	1.0012	1.0636	1.0663	**0.9992**	1.0385
	ResNext 29 8x16	0.9398	0.8224	0.8407	0.9218	0.8133	0.9011	0.8938	**0.8072**	0.8238
	VGG 19	1.5414	1.1998	1.2064	1.2993	1.1947	1.2158	1.2194	**1.1799**	1.2454
	WRN 28x10	0.8173	0.8138	0.8334	0.9257	0.7873	0.8600	0.8242	**0.7736**	0.8644
	WRN 40x10	1.2248	0.9058	0.9259	0.9883	0.8997	0.9533	0.9309	**0.8843**	1.1977
	Avg. Relative $\overline{NLL}$	1.0000	0.8770	0.8937	0.9742	0.8668	0.9329	0.9190	**0.8568**	0.9522

Models are denoted by their architecture and depth (and width if applicable)

Bold would be the best performing model

Table 12 Brier Score using 1000 samples

	Task	Uncalibrated	TS	ETS	BTS	HTS	LTS	HnLTS	PTS	PTS (L_{ECE})
CIFAR 10	DenseNet 121	0.0764	0.0729	0.0726	0.0755	0.0727	0.0730	0.0732	**0.0725**	0.0782
	DenseNet 169	0.0754	0.0718	0.0713	0.0735	0.0714	0.0719	0.0718	**0.0712**	0.0777
	ResNet 50	0.2392	0.2037	0.2030	0.2110	0.2034	0.2030	**0.2026**	0.2032	0.2281
	ResNet 101	0.1102	0.1011	**0.1011**	0.1069	0.1012	0.1011	0.1012	0.1011	0.1101
	ResNext 29 8x16	0.0828	**0.0784**	0.0784	0.0838	0.0786	0.0793	0.0796	0.0790	0.0853
	VGG 19	0.1101	0.1019	0.1008	0.1053	0.1009	0.1017	0.1011	**0.1007**	0.1131
	WRN 28x10	0.0629	**0.0608**	0.0610	0.0638	0.0609	0.0609	0.0614	0.0609	0.0644
	WRN 40x10	0.0820	0.0768	**0.0766**	0.0823	0.0770	0.0774	0.0775	0.0771	0.0851
	Avg. Relative $\overline{Brier}$	1.0000	0.9317	**0.9286**	0.9754	0.9303	0.9340	0.9347	0.9300	1.0162
CIFAR 100	DenseNet 121	0.3171	0.3048	0.3046	0.3095	0.3047	0.3130	0.3126	**0.3024**	0.3117
	DenseNet 169	0.3142	0.3017	0.3013	0.3054	0.3014	0.3087	0.3085	**0.2986**	0.3055
	ResNet 101	0.4053	0.3818	0.3818	0.3874	0.3820	0.3908	0.3908	**0.3810**	0.3867
	ResNext 29 8x16	0.3275	0.3096	0.3095	0.3142	0.3093	0.3197	0.3201	**0.3072**	0.3096
	VGG 19	0.4433	0.3920	0.3911	0.3917	0.3904	0.3950	0.3943	**0.3865**	0.3946
	WRN 28x10	0.2892	0.2887	0.2880	0.2905	0.2871	0.2919	0.2900	**0.2836**	0.2935
	WRN 40x10	0.3700	0.3278	0.3273	0.3345	0.3295	0.3317	0.3312	**0.3250**	0.3475
	Avg. Relative $\overline{Brier}$	1.0000	0.9397	0.9384	0.9508	0.9387	0.9581	0.9567	**0.9304**	0.9570

Models are denoted by their architecture and depth (and width if applicable)

Bold would be the best performing model

Table 13 ECE ($M = 50$) using 500 samples

	Task	Uncalibrated	TS	ETS	BTS	HTS	LTS	HnLTS	PTS	PTS (L_{ECE})
CIFAR 10	DenseNet 121	2.84	1.55	2.36	2.94	1.54	**1.33**	1.76	1.90	2.99
	DenseNet 169	2.79	1.59	2.41	2.58	1.44	**1.32**	1.66	1.87	3.14
	ResNet 50	10.71	2.69	2.76	5.09	**2.48**	2.56	2.58	2.53	9.71
	ResNet 101	4.45	**1.53**	1.82	3.96	1.75	1.77	2.04	2.18	4.42
	ResNext 29 8x16	2.88	**1.21**	1.22	3.29	1.42	1.55	1.89	2.23	3.35
	VGG 19	4.61	2.58	3.29	3.88	**2.06**	2.07	2.28	2.60	4.65
	WRN 28x10	1.93	**0.90**	1.80	2.37	0.97	0.95	1.40	1.62	2.37
	WRN 40x10	3.12	**1.13**	1.55	3.23	1.28	1.34	1.58	2.03	3.48
	Avg. Relative $\overline{ECE}$	1.00	0.44	0.62	0.95	0.44	**0.44**	0.54	0.61	1.07
CIFAR 100	DenseNet 121	8.76	4.04	**3.30**	4.94	3.44	7.55	8.12	3.54	8.26
	DenseNet 169	8.93	4.09	**3.19**	4.89	3.41	7.62	8.59	3.54	8.31
	ResNet 101	11.45	2.29	2.48	4.62	2.51	6.31	7.24	**2.23**	5.17
	ResNext 29 8x16	9.69	3.34	2.99	4.99	**2.52**	7.62	8.86	2.53	6.31
	VGG 19	17.63	5.08	5.33	5.27	4.07	5.56	6.50	**3.82**	6.62
	WRN 28x10	5.19	4.73	3.80	4.77	3.63	6.65	7.00	**3.54**	6.78
	WRN 40x10	14.78	4.12	**3.33**	5.30	3.81	6.88	6.67	4.03	9.67
	Avg. Relative $\overline{ECE}$	1.00	0.42	0.36	0.52	0.35	0.73	0.80	**0.35**	0.76

Models are denoted by their architecture and depth (and width if applicable)

Bold would be the best performing model

Table 14 NLL using 500 samples

	Task	Uncalibrated	TS	ETS	BTS	HTS	LTS	HnLTS	PTS	PTS (L_{ECE})
CIFAR 10	DenseNet 121	0.1881	**0.1623**	0.1690	0.4336	0.1625	0.1765	0.2032	0.2051	0.2180
	DenseNet 169	0.1870	**0.1617**	0.1686	0.4080	0.1623	0.1728	0.1969	0.2225	0.2386
	ResNet 50	0.7897	0.4481	0.4515	0.9523	**0.4474**	0.4529	0.4545	0.4740	0.9776
	ResNet 101	0.3047	**0.2170**	0.2212	0.5993	0.2184	0.2310	0.2476	0.2664	0.3541
	ResNext 29 8x16	0.1997	**0.1640**	0.1646	0.5475	0.1663	0.1895	0.2304	0.2754	0.2628
	VGG 19	0.2998	0.2362	0.2433	0.5869	**0.2348**	0.2468	0.2825	0.2895	0.3802
	WRN 28x10	0.1497	**0.1367**	0.1438	0.3427	0.1378	0.1463	0.1757	0.1835	0.1813
	WRN 40x10	0.2068	**0.1638**	0.1679	0.5126	0.1651	0.1803	0.1989	0.2361	0.2742
	Avg. Relative $\overline{NLL}$	1.0000	**0.7902**	0.8132	2.1410	0.7938	0.8519	0.9691	1.0584	1.2445
CIFAR 10	DenseNet 121	0.8939	0.8367	0.8604	1.0944	0.8268	1.0817	1.1304	**0.8238**	0.9738
	DenseNet 169	0.8748	0.8164	0.8346	1.0912	0.8075	1.0688	1.1528	**0.8040**	1.0126
	ResNet 101	1.1343	**1.0012**	1.0117	1.2453	1.0017	1.2018	1.2464	1.0022	1.0918
	ResNext 29 8x16	0.9398	0.8234	0.8410	1.1430	0.8146	1.1242	1.2370	**0.8124**	0.9120
	VGG 19	1.5414	1.2004	1.2079	1.4489	1.1955	1.2952	1.3199	**1.1796**	1.2775
	WRN 28x10	0.8173	0.8147	0.8335	1.0770	0.7883	0.9975	1.0226	**0.7791**	0.8900
	WRN 40x10	1.2248	0.9065	0.9315	1.2020	0.9007	1.0961	1.0846	**0.8899**	1.2474
	Avg. Relative $\overline{NLL}$	1.0000	0.8777	0.8953	1.1464	0.8676	1.0919	1.1415	**0.8619**	1.0166

Models are denoted by their architecture and depth (and width if applicable)

Bold would be the best performing model

Table 15 Brier Score using 500 samples

	Task	Uncalibrated	TS	ETS	BTS	HTS	LTS	HnLTS	PTS	PTS (L_{ECE})
CIFAR 10	DenseNet 121	0.0764	0.0729	**0.0727**	0.0790	0.0728	0.0736	0.0745	0.0751	0.0778
	DenseNet 169	0.0754	0.0718	**0.0713**	0.0770	0.0715	0.0724	0.0729	0.0738	0.0777
	ResNet 50	0.2392	0.2040	**0.2032**	0.2228	0.2038	0.2045	0.2043	0.2061	0.2394
	ResNet 101	0.1102	**0.1012**	0.1013	0.1135	0.1014	0.1022	0.1027	0.1042	0.1114
	ResNext 29 8x16	0.0828	**0.0786**	0.0786	0.0895	0.0788	0.0807	0.0821	0.0838	0.0862
	VGG 19	0.1101	0.1020	**0.1009**	0.1110	0.1010	0.1026	0.1027	0.1043	0.1118
	WRN 28x10	0.0629	**0.0608**	0.0611	0.0668	0.0609	0.0615	0.0627	0.0637	0.0649
	WRN 40x10	0.0820	**0.0768**	0.0769	0.0878	0.0769	0.0781	0.0787	0.0814	0.0849
	Avg. Relative $\overline{Brier}$	1.0000	0.9323	**0.9303**	1.0301	0.9314	0.9429	0.9511	0.9671	1.0232
CIFAR 100	DenseNet 121	0.3171	0.3051	0.3051	0.3151	0.3051	0.3294	0.3313	**0.3040**	0.3209
	DenseNet 169	0.3142	0.3018	0.3015	0.3109	0.3016	0.3259	0.3295	**0.2997**	0.3202
	ResNet 101	0.4053	0.3818	0.3820	0.3929	0.3820	0.4056	0.4071	**0.3816**	0.3923
	ResNext 29 8x16	0.3275	0.3098	0.3099	0.3219	0.3095	0.3387	0.3433	**0.3081**	0.3204
	VGG 19	0.4433	0.3920	0.3913	0.3958	0.3903	0.4051	0.4043	**0.3861**	0.3966
	WRN 28x10	0.2892	0.2888	0.2882	0.2952	0.2872	0.3039	0.3048	**0.2843**	0.2960
	WRN 40x10	0.3700	0.3277	0.3276	0.3421	0.3295	0.3455	0.3455	**0.3255**	0.3547
	Avg. Relative $\overline{Brier}$	1.0000	0.9399	0.9393	0.9677	0.9391	1.0013	1.0066	**0.9327**	0.9792

Models are denoted by their architecture and depth (and width if applicable)

Bold would be the best performing model

Table 16 ECE ($M = 50$) using 200 samples

	Task	Uncalibrated	TS	ETS	BTS	HTS	LTS	HnLTS	PTS	PTS (L_{ECE})
CIFAR 10	DenseNet 121	2.84	**1.72**	2.24	4.01	1.77	1.96	3.00	3.69	3.08
	DenseNet 169	2.79	1.70	2.86	3.99	**1.56**	1.74	2.66	3.62	3.05
	ResNet 50	10.71	2.69	3.23	7.88	**2.51**	2.92	3.19	5.47	10.04
	ResNet 101	4.45	**1.81**	2.16	5.11	2.03	2.26	3.11	4.45	4.67
	ResNext 29 8x16	2.88	**1.34**	1.72	4.12	1.51	1.87	3.00	3.95	3.04
	VGG 19	4.61	2.59	3.16	5.42	**2.40**	2.83	3.77	4.96	4.96
	WRN 28x10	1.93	**1.10**	2.24	3.56	1.23	1.46	2.46	2.73	2.43
	WRN 40x10	3.12	**1.39**	1.89	4.08	1.62	1.92	2.96	3.77	3.37
	Avg. Relative $\overline{ECE}$	1.00	**0.49**	0.71	1.31	0.51	0.59	0.89	1.15	1.08
CIFAR 100	DenseNet 121	8.76	4.01	**3.44**	8.78	3.49	19.00	17.72	4.77	9.01
	DenseNet 169	8.93	4.22	**3.44**	8.56	3.68	18.93	17.19	5.19	9.76
	ResNet 101	11.45	**2.77**	3.17	8.59	3.11	17.43	19.18	3.98	9.94
	ResNext 29 8x16	9.69	3.39	3.23	8.55	**2.73**	19.13	18.16	3.55	7.13
	VGG 19	17.63	5.16	5.53	8.46	**4.41**	14.55	15.33	4.48	11.71
	WRN 28x10	5.19	4.88	4.18	7.93	**4.01**	13.85	15.35	5.34	8.12
	WRN 40x10	14.78	4.52	**3.83**	9.64	4.47	15.71	17.47	5.23	9.87
	Avg. Relative $\overline{ECE}$	1.00	0.44	0.40	0.89	**0.38**	1.76	1.79	0.50	0.95

Models are denoted by their architecture and depth (and width if applicable)

Bold would be the best performing model

Table 17 NLL using 200 samples

	Model	Uncalibrated	TS	ETS	BTS	HTS	LTS	HnLTS	PTS	PTS (L_{ECE})
CIFAR 10	DenseNet 121	0.1881	**0.1654**	0.1709	0.6331	0.1671	0.2511	0.3918	0.4297	0.2263
	DenseNet 169	0.1870	**0.1633**	0.1748	0.5877	0.1646	0.2285	0.3335	0.4008	0.2342
	ResNet 50	0.7897	**0.4489**	0.4553	1.4653	0.4489	0.5041	0.5241	∞	0.8810
	ResNet 101	0.3047	**0.2199**	0.2248	0.7817	0.2256	0.2811	0.4023	0.5189	0.3902
	ResNext 29 8x16	0.1997	**0.1668**	0.1734	0.6465	0.1698	0.2523	0.3968	0.4850	0.2282
	VGG 19	0.2998	**0.2377**	0.2422	0.8606	0.2385	0.3304	0.4858	0.5694	0.3981
	WRN 28x10	0.1497	**0.1381**	0.1476	0.4952	0.1457	0.1955	0.2971	0.2930	0.1830
	WRN 40x10	0.2068	**0.1662**	0.1746	0.6286	0.1766	0.2525	0.3927	0.4728	0.2533
	Avg. Relative $\overline{NLL}$	1.0000	**0.7997**	0.8329	2.9233	0.8188	0.1263	1.6678	2.1004	1.2213
CIFAR 100	DenseNet 121	0.8939	0.8374	0.8595	1.7011	**0.8285**	2.2353	2.3776	0.9091	1.0147
	DenseNet 169	0.8748	0.8181	0.8359	1.6978	**0.8089**	2.1935	2.2927	0.9039	0.9845
	ResNet 101	1.1343	**1.0036**	1.0147	1.8572	1.0059	2.3447	2.7525	1.0721	1.2336
	ResNext 29 8x16	0.9398	0.8248	0.8420	1.6495	**0.8174**	2.3144	2.4890	0.8691	0.9635
	VGG 19	1.5414	1.2013	1.2099	1.8499	**1.1977**	2.0490	2.2260	1.2177	1.6701
	WRN 28x10	0.8173	0.8168	0.8362	1.5552	**0.7920**	1.6668	2.0215	0.9093	0.9380
	WRN 40x10	1.2248	0.9073	0.9323	1.9061	**0.9046**	2.0570	2.4555	0.9438	1.2121
	Avg. Relative $\overline{NLL}$	1.0000	0.8791	0.8967	1.6993	**0.8704**	2.0837	2.3254	0.9419	1.0849

Models are denoted by their architecture and depth (and width if applicable)

Bold would be the best performing model

Table 18 Brier Score using 200 samples

	Task	Uncalibrated	TS	ETS	BTS	HTS	LTS	HnLTS	PTS	PTS (L_{ECE})
CIFAR 10	DenseNet 121	0.0764	0.0734	**0.0733**	0.0852	0.0734	0.0766	0.0805	0.0841	0.0781
	DenseNet 169	0.0754	0.0719	0.0720	0.0849	**0.0718**	0.0746	0.0780	0.0833	0.0774
	ResNet 50	0.2392	0.2038	0.2039	0.2409	**0.2037**	0.2092	0.2093	∞	0.2379
	ResNet 101	0.1102	**0.1016**	0.1019	0.1200	0.1020	0.1051	0.1082	0.1161	0.1129
	ResNext 29 8x16	0.0828	**0.0789**	0.0801	0.0946	0.0792	0.0834	0.0884	0.0939	0.0844
	VGG 19	0.1101	0.1025	**0.1012**	0.1196	0.1019	0.1065	0.1099	0.1166	0.1133
	WRN 28x10	0.0629	**0.0610**	0.0617	0.0742	0.0611	0.0638	0.0675	0.0694	0.0651
	WRN 40x10	0.0820	**0.0772**	0.0781	0.0917	0.0776	0.0816	0.0859	0.0906	0.0842
	Avg. Relative $\overline{Brier}$	1.0000	**0.9357**	0.9393	1.1080	0.9363	0.9758	1.0167	1.0946	1.0225
CIFAR 100	DenseNet 121	0.3171	0.3052	0.3053	0.3355	**0.3049**	0.4109	0.3948	0.3118	0.3238
	DenseNet 169	0.3142	0.3022	**0.3019**	0.3306	0.3020	0.4073	0.3876	0.3098	0.3224
	ResNet 101	0.4053	**0.3824**	0.3830	0.4161	0.3827	0.4813	0.4842	0.3889	0.4096
	ResNext 29 8x16	0.3275	0.3101	0.3103	0.3402	**0.3097**	0.4215	0.4079	0.3130	0.3244
	VGG 19	0.4433	0.3921	0.3917	0.4118	**0.3907**	0.4588	0.4556	0.3911	0.4246
	WRN 28x10	0.2892	0.2892	0.2891	0.3120	**0.2877**	0.3557	0.3583	0.2954	0.2996
	WRN 40x10	0.3700	**0.3282**	0.3284	0.3667	0.3306	0.4107	0.4160	0.3327	0.3544
	Avg. Relative $\overline{Brier}$	1.0000	0.9409	0.9410	1.0250	**0.9404**	1.2059	1.1872	0.9554	1.0000

Models are denoted by their architecture and depth (and width if applicable)

Bold would be the best performing model

Results of ECE using 50 bins ($M = 50$), NLL, and Brier score, using the whole set are shown in TableS 4, 5, and 6, respectively. Tables 7, 8, and 9, show average results using 5000 samples, randomly chosen at each experiment run, to calibrate models. Equivalently, Tables 10, 11, and 12, show the same results but using 1000 samples; Tables 13, 14, and 15 show average results using 500 samples. Lastly, Tables 16, 17, and 18 show average results using 200 samples.

Acknowledgements D. R. is funded by grant PID2021-125943OB-I00, MCIN /AEI /10.13039/501100011033 / FEDER, UE from the Spanish Ministerio de Ciencia e Innovación, Agencia y del Fondo Europeo de Desarrollo Regional. Juan Maroñas acknowledges funding coming from the Spanish National Research Project PID2019-106827GB-I00, and from the Autonomous Community of Madrid (ELLIS Unit Madrid).

Author Contributions SA.B contributed to Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Data Curation, Writing - Original Draft, Writing - Review & Editing, Visualization. JM contributed to Methodology, Validation, Formal analysis, Investigation, Resources, Writing - Review & Editing, Supervision. DR contributed to Methodology, Validation, Formal analysis, Resources, Writing - Review & Editing, Supervision, Project administration, Funding acquisition.

Funding Open Access funding provided thanks to the CRUE-CSIC agreement with Springer Nature.

Data Availability All data generated or analyzed during this study are included in this published article [41].

Declarations

Conflict of interest The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Ding Z, Han X, Niethammer M (2020) Votenet+ : an improved deep learning label fusion method for multi-atlas segmentation. In: Proceedings IEEE international symposium on biomedical imaging pp. 363–367. https://doi.org/10.1109/isbi45749.2020.9098493
2. Hinton G, Deng L, Yu D, Dahl GE, Mohamed A-R, Jaitly N, Senior A, Vanhoucke V, Nguyen P, Sainath TN, Kingsbury B (2012) Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups. IEEE Signal Process Mag 29(6):82–97. https://doi.org/10.1109/MSP.2012.2205597
3. Duda RO (2001) Pattern classification, 2nd, ed. Wiley, New York

4. Dawid AP (1982) The well-calibrated Bayesian. J Am Stat Assoc 77(379):605–610. https://doi.org/10.1080/01621459.1982.10477856
5. DeGroot MH, Fienberg SE (1983) The comparison and evaluation of forecasters. J R Stat Soc Ser D 32(1–2):12–22. https://doi.org/10.2307/2987588
6. Guo C, Pleiss G, Sun Y, Weinberger KQ (2017) On calibration of modern neural networks. In: 34th International conference on machine learning ICML 3: 2130–2143 arXiv:1706.04599
7. Ovadia Y, Fertig E, Ren J, Nado Z, Sculley D, Nowozin S, Dillon JV, Lakshminarayanan B, Snoek J (2019) Can you trust your model's uncertainty? Evaluating predictive uncertainty under dataset shift. Adv Neural Inform Process Syst 32(NeurIPS) arXiv:1906.02530
8. Lakshminarayanan B, Pritzel A, Blundell C (2017) Simple and scalable predictive uncertainty estimation using deep ensembles. In: Proceedings of the 31st international conference on neural information processing systems. NIPS'17, pp. 6405–6416. Curran Associates Inc., Red Hook, NY, USA
9. Shabat TB, Meir R, Azriel D (2022) Empirical bayes approach to truth discovery problems. In: The 38th conference on uncertainty in artificial intelligence
10. Kristiadi A, Hein M, Hennig P (2020) Being Bayesian, even just a bit, fixes overconfidence in ReLU networks. In: 37th International conference on machine learning ICML 2020, vol. PartF16814, pp. 5392–5402
11. Singh Sambyal A, Krishnan NC, Bathula DR (20222) toward reducing aleatoric uncertainty for medical imaging tasks. In: 2022 IEEE 19th International symposium on biomedical imaging (ISBI), pp. 1–4. https://doi.org/10.1109/ISBI52829.2022.9761638
12. Patel K, Beluch W, Zhang D, Pfeiffer M, Yang B (2020) On-manifold adversarial data augmentation improves uncertainty calibration. In: Proceedings—International conference on pattern recognition, pp. 8029–8036. https://doi.org/10.1109/ICPR48806.2021.9413010
13. Hendrycks D, Mu N, Cubuk ED, Zoph B, Gilmer J, Lakshminarayanan B (2020) AugMix: a simple data processing method to improve robustness and uncertainty. In: Proceedings of the international conference on learning representations (ICLR)
14. Karandikar A, Cain N, Tran D, Lakshminarayanan B, Shlens J, Mozer MC, Roelofs B (2021) Soft calibration objectives for neural networks. Adv Neural Inform Process Syst 6(34):29768–79
15. Tomani C, Cremers D, Buettner F (2021) Parameterized temperature scaling for boosting the expressive power in post-hoc uncertainty calibration
16. Bohdal O, Yang Y, Hospedales T (2023) Meta-calibration: learning of model calibration using differentiable expected calibration error. Trans Mach Learning Res. https://doi.org/10.48550/arXiv.2106.09613
17. Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, Killeen T, Lin Z, Gimelshein N, Antiga L, Desmaison A, Kopf A, Yang E, DeVito Z, Raison M, Tejani A, Chilamkurthy S, Steiner B, Fang L, Bai J, Chintala S (2019) Pytorch: an imperative style, high-performance deep learning library. In: Proceedings of the 33rd international conference on neural information processing systems. Curran Associates Inc., Red Hook, NY, USA
18. Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C, Corrado GS, Davis A, Dean J, Devin M, Ghemawat S, Goodfellow I, Harp A, Irving G, Isard M, Jia Y, Jozefowicz R, Kaiser L, Kudlur M, Levenberg J, Mané D, Monga R, Moore S, Murray D, Olah C, Schuster M, Shlens J, Steiner B, Sutskever I, Talwar K, Tucker P, Vanhoucke V, Vasudevan V, Viégas F, Vinyals O, Warden P, Wattenberg M, Wicke M, Yu Y, Zheng X (2015) TensorFlow: large-scale machine learning on heterogeneous systems. Software available from tensorflow.org. https://www.tensorflow.org/
19. Maroñas J, Paredes R, Ramos D (2020) Calibration of deep probabilistic models with decoupled Bayesian neural networks. Neurocomputing 407:194–205
20. Wenger J, Kjellström H, Triebel R (2020) Non-parametric calibration for classification. In: International conference on artificial intelligence and statistics (AISTATS) pp. 178–190
21. Minderer M, Djolonga J, Romijnders R, Hubis F, Zhai X, Houlsby N, Tran D, Lucic M (2021) Revisiting the calibration of modern neural networks. Adv Neural Inf Proc Syst 34: 15682–15694
22. Zhang J, Kailkhura B, Han T (2020) Mix-n-match: ensemble and compositional methods for uncertainty calibration in deep learning. In: International conference on machine learning (ICML)
23. Rahimi A, Shaban A, Cheng CA, Hartley R, Boots B (2020) Intra order-preserving functions for calibration of multi-class neural networks. In: Advances in neural information processing systems, vol. 2020
24. Sadjadi S, Kheyrkhah T, Tong A, Greenberg C, Reynolds D, Singer E, Mason L, Hernandez-Cordero J (2018) The 2017 nist language recognition evaluation, pp. 82–89 (2018). https://doi.org/10.21437/Odyssey.2018-12
25. Brummer N, Van Leeuwen DA (2006) On calibration of language recognition scores. In: 2006 IEEE Odyssey - the speaker and language recognition workshop pp. 1–8. https://doi.org/10.1109/ODYSSEY.2006.248106
26. Yoo TK, Choi JY, Kim HK (2021) Feasibility study to improve deep learning in OCT diagnosis of rare retinal diseases with few-shot classification. Med Biol Eng Compu 59(2):401–415. https://doi.org/10.1007/s11517-021-02321-1
27. Mitchell TM (1980) The need for biases in learning generalizations
28. Lecun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. Proc IEEE 86(11):2278–2324. https://doi.org/10.1109/5.726791
29. Ding Z, Han X, Liu P, Niethammer M (2020) Local Temperature scaling for probability calibration
30. Naeini MP, Cooper GF, Hauskrecht M (2015) Obtaining well calibrated probabilities using Bayesian Binning. In: Proceedings of the national conference on artificial intelligence, vol. 4, pp. 2901–2907
31. Kumar A, Liang PS, Ma T (2019) Advances in Neural Information Processing Systems. Curran Associates Inc, US. https://proceedings.neurips.cc/paper_files/paper/2019/file/f8c0c968632845cd133308b1a494967f-Paper.pdf
32. Vaicenavicius J, Lindsten F, Widmann D, Roll J, Andersson C, Schön TB (2020) Evaluating model calibration in classification. In: AISTATS 2019 - 22nd international conference on artificial intelligence and statistics
33. Nixon J, Dusenberry M, Jerfel G, Nguyen T, Liu J, Zhang L, Tran D (2019) Measuring calibration in deep learning. In: Proceedings - 2019 IEEE conference on computer vision and pattern recognition workshops, pp. 38–41
34. Simonoff JS, Udina F (1997) Measuring the stability of histogram appearance when the anchor position is changed. Comput Stat Data Anal 23(3):335–353. https://doi.org/10.1016/S0167-9473(96)00040-0
35. Bröcker J (2009) Reliability, sufficiency, and the decomposition of proper scores. Q J R Meteorol Soc 135(643):1512–1519
36. Dawid AP, Musio M (2014) Theory and applications of proper scoring rules. Metron 72(2):169–183. https://doi.org/10.1007/s40300-014-0039-y

37. Brümmer N (2010) Measuring, refining and calibrating speaker and language information extracted from speech. PhD thesis, Stellenbosch University
38. Pereyra G, Tucker G, Chorowski J, Kaiser Ł, Hinton G (2019) Regularizing neural networks by penalizing confident output distributions. In: 5th International conference on learning representations, ICLR 2017 - workshop track proceedings. https://doi.org/10.48550/arxiv.1701.06548.https://arxiv.org/abs/1701.06548
39. Ji B, Jung H, Yoon J, Kim K, Shin Y (2019) Bin-wise temperature scaling (BTS): improvement in confidence calibration performance through simple scaling techniques. In: Proceedings - 2019 international conference on computer vision workshop, ICCVW 2019, pp. 4190–4196. https://doi.org/10.1109/ICCVW.2019.00515
40. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE computer society conference on computer vision and pattern recognition 2016, 770–778 arXiv:1512.03385. https://doi.org/10.1109/CVPR.2016.90
41. Krizhevsky A, Hinton G (2009) Learning multiple layers of features from tiny images. Technical report
42. Sutskever I, Martens J, Dahl G, Hinton G (2013) On the importance of initialization and momentum in deep learning. In: Proceedings of the 30th international conference on machine learning, vol. 28, pp. 1139–1147. PMLR. https://proceedings.mlr.press/v28/sutskever13.html
43. ...Virtanen P, Gommers R, Oliphant TE, Haberland M, Reddy T, Cournapeau D, Burovski E, Peterson P, Weckesser W, Bright J, van der Walt SJ, Brett M, Wilson J, Millman KJ, Mayorov N, Nelson ARJ, Jones E, Kern R, Larson E, Carey CJ, Polat İ, Feng Y, Moore EW, VanderPlas J, Laxalde D, Perktold J, Cimrman R, Henriksen I, Quintero EA, Harris CR, Archibald AM, Ribeiro AH, Pedregosa F, van Mulbregt P (2020) SciPy 1.0 contributors: SciPy 1.0: fundamental algorithms for scientific computing in python. Nat Methods 17:261–272. https://doi.org/10.1038/s41592-019-0686-2
44. Huang G, Liu Z, Van Der Maaten L, Weinberger KQ (2017) Densely connected convolutional networks. In: Proceedings - 30th IEEE conference on computer vision and pattern recognition, CVPR 2017 2017, 2261–2269 arXiv:1608.06993. https://doi.org/10.1109/CVPR.2017.243