Post-hoc Calibration of Neural Networks

Amir Rahimi
ANU

Kartik Gupta
ANU & Data61, CSIRO

Thalaiyasingam Ajanthan
ANU

Thomas Mensink
Google Research

Cristian Sminchisescu
Google Research

Richard Hartley
ANU & Google Research

Abstract

Calibration of neural networks is a critical aspect to consider when incorporating machine learning models in real-world decision-making systems where the confidence of decisions are equally important as the decisions themselves. In recent years, there is a surge of research on neural network calibration and the majority of the works can be categorized into *post-hoc* calibration methods, defined as methods that learn an additional function to calibrate an already trained base network. In this work, we intend to understand the post-hoc calibration methods from a theoretical point of view. Especially, it is known that minimizing Negative Log-Likelihood (NLL) will lead to a calibrated network on the training set if the global optimum is attained [1]. Nevertheless, it is not clear learning an additional function in a post-hoc manner would lead to calibration in the theoretical sense. To this end, we prove that even though the base network (f) does not lead to the global optimum of NLL, by adding additional layers (g) and minimizing NLL by optimizing the parameters of g one can obtain a calibrated network $g \circ f$. This not only provides a less stringent condition to obtain a calibrated network but also provides a theoretical justification of post-hoc calibration methods. Our experiments on various image classification benchmarks confirm the theory.

1 Introduction

In this paper we consider the problem of calibration of neural networks, or classification functions in general. This problem has been considered in the context of Support Vector Machines [17], but has recently been considered in the context of Convolutional Neural Networks (CNNs) [4]. In this case, a CNN used for classification takes an input $x \in \mathcal{D}_X$, belonging to one of n classes, and outputs a vector $f(x)$ in $\mathbb{R}^n$, where the k-th component, $f_k(x)$ is often interpreted as a probability that input x belongs to class k. If this value is to represent probabilities accurately, then we require that $f_k(x) = P(k \mid f(x))$. In this case, the classifier f is said to be *calibrated*, or *multi-class calibrated.* [1]

A well-known condition ([1]) for a classifier to be calibrated is that it minimizes the *cross-entropy* cost function, over all functions $f : \mathcal{D}_X \to \Delta^{n-1}$, where Δ^{n-1} is the standard probability simplex (see definition below). If the absolute minimum is attained, it is true that $f_k(x) = P(k \mid x)$. However, this condition is rarely satisfied, since $\mathcal{D}_X$ may be a very large space (for instance a set of images, of very high dimension) and the task of finding the absolute (or even a local) minimum of the loss is difficult: it requires the network to have sufficient capacity, and also that the network manages to find the optimal value through training. To fulfil this requirement, two networks that reach different minima of the loss function cannot both be calibrated. However, the requirement that a network be calibrated is separate from that of finding the optimal classifier.

[1] In many papers, *e.g.* [11] and calibration metrics, *e.g.* ECE [15] a slightly different condition known as *classwise-calibration* is preferred: $f_k(x) = P(k \mid f_k(x))$.

In this paper, it is shown that a far less stringent condition is sufficient for the network to be calibrated; we say that the network is *optimal with respect to calibration* or *calibration-optimal* provided no adjustment of the output of the network *in the output space* can improve the calibration (see definition 3.3). This is a far simpler problem, since it requires that a function between far smaller-dimensional spaces should be optimal.

Achieving optimality with respect to calibration can be achieved in either of two ways:

1. by addition of extra layers at the end of the network and post-hoc training on a hold-out calibration set to minimize the *cross-entropy* cost function. The extra layers (which we call g-layers) can be added before or after the usual softmax layer. Since the output space of the network is of small dimension (compared to the input of the whole network), optimization of the loss by training the g-layers is a far easier task;
2. by training the network with the small g-layers in place from the beginning, on the training set. Since the g-layers are small, and near the end of the network, the expectation is that they will become trained far more easily and more quickly than the rest of the network. A final training of the g-layers alone on a small hold-out calibration set should also be applied.

We conduct experiments on various image classification datasets by learning a small fully-connected network for the g-layers on a hold-out calibration set and evaluate on an unseen test set. Our experiments confirm the theory that if the calibration set and the test set are statistically similar, our method outperforms existing post-hoc calibration methods while retaining the original accuracy.

2 Preliminaries

We consider a pair of joint random variables, (X, K). Random variable X should take values in some domain $\mathcal{D}_X$ for instance a set of images, and K takes values in a finite class set $\mathcal{K} = \{1, 2, \ldots, n\}$. The variable n will refer always to the number of labels, and k denotes an element of the class set.

We shall be concerned with a (measurable) function $f : \mathcal{D}_X \to \mathcal{D}_Z = \mathbb{R}^m$, and random variable Z defined by $Z = f(X)$. Note that $\mathcal{D}_Z$ is the same as $\mathbb{R}^m$, but we shall usually use the notation $\mathcal{D}_Z$ to remind us that it is the range of function f. The distribution of the random variable X induces the distribution for the random variable $Z = f(X)$. The symbol z will always represent $f(x)$ where x is a value of random variable X. The notation $x \sim X$ means that x is a value sampled from the random variable X. The situation we have in mind is that f is the function implemented by a (convolutional) neural network. A notation $P(\cdot)$ (with upper-case P) always refers to probability, whereas a lower case p represents a probability distribution. We use the notation $P(k \mid z)$ for brevity to mean $P(K = k \mid Z = z)$.

A common way of doing classification, given n classes, is that the neural net is terminated with a layer represented by a function $q : \mathbb{R}^m \to \mathbb{R}^n$ (where typically $m = n$, but this is not required), taking value $q(z) = (q_1(z), \ldots, q_n(z))$ in $\mathbb{R}^n$, and satisfying $q_k(z) > 0$ and $\sum_{k=1}^n q_k(z) = 1$. The set of such vectors $q(z)$ satisfying these conditions is called the *standard probability simplex*, Δ^{n-1}, or simply the standard (open) simplex. This is an $n-1$ dimensional subset of $\mathbb{R}^n$. An example of such a function q is the softmax function defined by $q_k(z) = \exp(z_k) / \sum_{j=1}^n \exp(z_j)$.

Thus, the function implemented by a neural net is $q \circ f$, where $f : \mathcal{D}_X \to \mathcal{D}_Z = \mathbb{R}^m$, and $q : \mathbb{R}^m \to \Delta^{n-1}$. The function q will be called the *activation* in this paper. The notation $q \circ f$ represents the composition of the two functions f and q. One is tempted to declare (or hope) that $q_k(z) = P(k \mid z)$, in other words that the neural network outputs the correct conditional class probabilities given the network output. At least it is assumed that the most probable class assignment is equal to $\text{argmax}_{k \in \mathcal{K}}\, q_k(z)$. It will be investigated how justified these assumptions are. Clearly, since f can be any function, this is not going to be true in general.

2.1 Loss

Given a pair $(x, k) \in \mathcal{D}_X \times \mathcal{K}$, the negative-log loss is given by $L(q \circ f, x, k) = -\log(q_k(f(x)))$. The expected loss over the distribution given by the random variables (X, K) is

$$L(q \circ f, X, K) = E_{(x,k)\sim(X,K)} L(q \circ f, x, k) = -E_{(x,k)\sim(X,K)} \log(q_k(f(x))) \, . \qquad (1)$$

We cannot know the complete distribution of the random variables (X, K) in a real situation, however, if the distributions are represented by data pairs $\mathcal{D} = \{(x_i, k_i)_{i=1}^N\}$ sampled from the distribution of (X, K), then the expected loss is approximated by the empirical loss

$$L(q \circ f, \mathcal{D}) \approx -E_{(x,k)\sim\mathcal{D}} \log(q_k(f(x)) = -\sum_{i=1}^{N} \log(q_{k_i}(f(x_i)) \ . \tag{2}$$

The training process of the neural network is intended to find the function f^* that minimizes the loss. Thus $f^* = \operatorname{argmin}_{f:\mathcal{D}_X \to \mathbb{R}^m} L(q \circ f, \mathcal{D})$.

3 Calibration

According to the theory (see [1]), if the trained network is $f^* = \operatorname{argmin}_{f:\mathcal{D}_X \to \Delta^{n-1}} L(f, X, K)$, where $L(f, X, K)$ is the loss given in (1) then the network (function f^*) is calibrated, in the sense that $f_k^*(x) = P(k \mid x)$, as stated in the following theorem.

Theorem 3.1. *Consider joint random variables (X, K), taking values in $\mathcal{D}_X$ and $\mathcal{K}$ respectively, where $\mathcal{D}_X$ is some Cartesian space. Let $f : \mathcal{D}_X \to \Delta^{n-1}$ be a function. Define the loss $L(f, X, K) = -E_{(x,k)\sim(X,K)} \log\big(f_k(x)\big)$. If $f = \operatorname{argmin}_{\hat{f}:\mathcal{D}_X \to \Delta^{n-1}} L(\hat{f}, X, K)$ then $P(k \mid x) = f_k(x)$.*

This theorem is a simple corollary of the following slight generalization, which is proved in the supplementary material. (The theorem is stated in terms of a function q, rather than f for convenience later.)

Theorem 3.2. *Consider joint random variables (Z, K), taking values in $\mathbb{R}^m$ and $\mathcal{K}$ respectively. Let $q : \mathbb{R}^m \to \Delta^{n-1}$ be a submersion.[2] Define the loss $L(q, Z, K) = -E_{(z,k)\sim(Z,K)} \log\big(q_k(z)\big)$. If*

$$\mathrm{id} = \operatorname*{argmin}_{g:\mathbb{R}^m \to \mathbb{R}^m} L(q \circ g, Z, K) \ ,$$

where $\mathrm{id} : \mathbb{R}^m \to \mathbb{R}^m$ is the identify function. Then $P(k \mid z) = q_k(z)$.

This theorem weakens the condition for optimality of f (or q) by the weaker requirement that it be optimal with respect to the prepended modification by $g : \mathbb{R}^m \to \mathbb{R}^m$. The theorem in this form will be used to derive our main theorem.

Theorem 3.1 is a fundamental result, but it leaves the following difficulties. Even if the network is trained to completion, or trained with early-stopping, there is no expectation that the loss will be exactly minimized over all possible functions $f : \mathcal{D}_X \to \Delta^{n-1}$. If this were always the case, then research into different network architectures would be largely superfluous. We define a less ambitious minimum which does not require the whole network to be optimal, as follows.

Definition 3.3. A function $f : \mathcal{D}_X \to \mathbb{R}^m$ is said to be *optimal with respect to calibration* for a loss-function $L(\cdot, X, K)$ and activation q if

$$\operatorname*{argmin}_{g:\mathbb{R}^m \to \mathbb{R}^m} L(q \circ g \circ f, X, K) = \mathrm{id} \ ,$$

where $\mathrm{id} : \mathbb{R}^m \to \mathbb{R}^m$ is the identify function.

In other words f is optimal with respect to calibration if replacing f by the *composite* function $g \circ f$ does not result in a decrease in the loss function.

Now, we can state and prove our theorem on calibration.

Theorem 3.4. *Consider joint random variables (X, K), taking values in $\mathcal{D}_X$ and $\mathcal{K}$ respectively. Let $f : \mathcal{D}_X \to \mathbb{R}^m$, and let $q : \mathbb{R}^m \to \Delta^{n-1}$ be a submersion. Define the loss*

$$L(q \circ f, X, K) = -E_{(x,k)\sim(X,K)} \log\big(q_k(f(x))\big) \ .$$

If f is optimal with respect to calibration, for this cost function, then $P(k \mid f(x)) = q_k(f(x))$.

[2]A function between two differential manifolds $q : \mathcal{M} \to \mathcal{N}$ is called a submersion if its differential map at point $z \in \mathcal{M}$, namely $Dq_z : T_z\mathcal{M} \to T_{q(z)}\mathcal{N}$ has rank equal to the dimension of $\mathcal{N}$.

This theorem also holds for least-squares error as indicated in [1] and various other cost functions generally known as proper loss functions [2, 19].

Proof. Let $Z = f(X)$, and for any x define $f(x) = z$. Then,

$$L(q \circ f, X, K) = -E_{(x,k)\sim(X,K)} \log\big(q_k(f(x))\big) = -E_{(z,k)\sim(Z,K)} \log\big(q_k(z)\big) = L(q, Z, K) \; .$$

According to Theorem 3.2, if replacing q by $q\circ g$ will not decrease the value of the loss function (which is the definition that q is optimal with respect to recalibration), we may conclude that $P(k \mid z) = q_k(z)$ as required. □

This theorem may be applied in the standard case where $m = n$ and q is the softmax function (which is a submersion). We give examples of other situations when this theorem applies.

Two-class classification. For two-class classification, we may select $\mathcal{D}_Z = \mathbb{R}$. Let h be any strictly increasing function with range $(0, 1)$. The sigmoid function is one such function, so are $\arctan(z)/\pi + 1/2$ and $(\tanh(z) + 1)/2$. Define $q_1(z) = h(z)$ and $q_2(z) = 1 - h(z)$. In this case, the Jacobian is $[h'(z), -h'(z)]$ which has rank 1, so h is a submersion. The function f is calibrated if there is no function $g : \mathbb{R} \rightarrow \mathbb{R}$ such that $g \circ f$ decreases the loss.

The main point here is that it is not necessary to find the function f that achieves the minimum of $L(f, X, K)$ for the function to be calibrated. All that is necessary is to minimize the cost over functions $g : \mathbb{R} \rightarrow \mathbb{R}$, or even to find a local minimum, such that small variations of f do not decrease the cost.

Multi-class classification. In a multiclass classification problem, we may choose $m = n$, and define functions $q_k : \mathbb{R}^m \rightarrow \mathbb{R}$ so that $q : \mathbb{R}^m \rightarrow \Delta^{m-1}$ is a submersion. For example, let h_k be a positive strictly increasing (or decreasing) function for $k = 1, \ldots, m$, and define $q_k(z) = h_k(z_k)/\sum_{j=1}^{n} h_j(z_j)$. The denominator is simply a normalization factor to ensure that $\sum_{k=1}^{n} q_k(z) = 1$. Together, all the functions q_k make up a function $q : \mathbb{R}^m \rightarrow \Delta^{m-1}$, and it may be verified that it is a submersion. An example of such a function is the softmax (when $h_k(z_k) = \exp(z_k)$).

Calibrating partially trained networks. According to Theorem 3.4, there is no need for the classifier network f_θ to be optimized in order for it to be calibrated. It is sufficient that the last layer of the network, (before the *softmax* layer, represented by q) should be optimal. Thus, it is possible for the classifier to be calibrated even after early-stopping or incomplete training.

Classwise calibration. Theorem 3.4 gives a condition for the network to be calibrated in the sense called *multi-class-calibration* in [11]. Many other calibration methods [13, 17, 21] aim at *classwise-calibration*. It can be shown (see the supplementary material) that if a classifier is correctly multi-class-calibrated, then it is classwise-calibrated, and furthermore it is calibrated correctly for the top-r prediction, or within-top-r predictions.[3] The converse does not hold.

4 Finding a calibrating CNN

A simple rewriting of Theorem 3.4 is

Corollary 4.5. *Consider joint random variables (X, K), taking values in $\mathcal{D}_X$ and $\mathcal{K}$ respectively. Let $f : \mathcal{D}_X \rightarrow \mathbb{R}^m$, and $Z = f(X)$. Further, let $q : \mathbb{R}^m \rightarrow \Delta^{n-1}$ be a submersion. If*

$$g = \underset{\hat{g}:\mathbb{R}^m\rightarrow\mathbb{R}^m}{\operatorname{argmin}} -E_{(z,k)\sim(Z,K)} \log\big(q_k(\hat{g}(z))\big) \; , \tag{3}$$

then $P(k \mid g \circ f(x)) = q_k(g \circ f(x))$.

Our strategy, therefore, is to replace function f by $g \circ f$, where g minimizes the loss function $L(q \circ g, Z, K)$ in (3). Then the function $g \circ f$ will be calibrated. It is assumed that f is implemented by a neural net, and so is g. Therefore, the method consists of steps, starting with neural net f_θ which has already been trained to compute function $f = f_\theta$:

[3]The network is said to be calibrated for within-top-r predictions if the probability of the correct answer being one of the top r predictions is equal to the sum of the top r scores.

1. Strip any softmax layer (or equivalent) from f, and capture samples $(z, k) \sim (f(X), K)$ from a *calibration* set, which should be different from the set used to train f.
2. Train a neural network g_ϕ on the captured samples to minimize (3), providing $g = g_\phi$.
3. The composite function neural network $g_\phi \circ f_\theta$ is the modified network.

According to Corollary 4.5, the output of the composite network $g \circ f$ will be calibrated, provided that the minimum is achieved when training g_ϕ and that the calibration dataset accurately represents the distribution $Z = f(X)$.

It is a far simpler task to train a network g_ϕ to minimize $L(q \circ g, Z, K)$ than it is to train f_θ to minimize $L(q \circ f, X, K)$, since the dimension of the data $Z \in \mathbb{R}^m$ is normally far smaller than the dimension of $\mathcal{D}_X$. In our experiments, we implement g as a small multilayer perceptron (MLP) consisting of up to 4 dense layers, of dimension no greater than a small multiple of m. Training time for g_ϕ is usually less than a minute.

Initialization. Assuming that function f_θ has already been trained to minimize the loss on the training set, when g_ϕ is trained we do not wish to undo all the work that has been done by starting training g_ϕ from an arbitrary point. Therefore, we initialize the parameters of g_ϕ so that initially it implements the identity function. We refer to these layers as *transparent layers* in their initial parametrization. This is similar to the approach in [3].

An alternative is to train $g_\phi \circ f_\theta$ on the training set, followed by a short period of training on the calibration set, keeping the parameters θ fixed. In this case, it is not necessary to initialize the g-layers to be transparent.

4.1 Overfitting

We observe that achieving very good calibration on the *calibration* set used to train the g-layers is relatively easy. Despite this, calibration as measured on the *test* set, although far better than the calibration of the original network f_θ, is not always as good.

The main reason for this is that *the calibration and test sets are statistically dissimilar*. Although fitting g-layers to minimize cross-entropy loss gives near-perfect calibration on the calibration set, this does not mean that it will be minimized on the test set. In other words, the g-layers are overfitted to the calibration set. This is similar to the observation that the original network is well calibrated on the training set but usually poorly calibrated on the test set.

The phenomenon of overfitting to the calibration set has been observed by many authors as far back as [17]. The lesson from this is that the set used for calibration of the g-layers should be relatively large. For the CIFAR-10 dataset, we used $45,000$ training samples and 5000 calibration samples (standard practice in calibration literature), but a different split of the data may provide better calibration results. In addition, the number of parameters in the g-layers should be kept low to avoid overfitting. For this reason, using more than 3 layers for g_ϕ seems counter-productive.

5 Related Work

Calibrating classification functions has been studied for the past few decades, earlier in the context of support vector machines [17, 21] and recently on neural networks [4]. In this literature, it is typically preferred to calibrate an already trained classifier, denoted as *post-hoc calibration*, as it can be applied to any off-the-shelf classifier. Over the past few years, many post-hoc calibration methods have been developed such as temperature scaling [4] for multi-class classification), Bayesian binning [15], beta calibration [12] and its extensions [11] to name a few. These methods are learned on a hold-out calibration set and the main difference among them is the type of function learned and the heuristics used to avoid overfitting to the calibration set. Specifically, temperature scaling learns a scalar parameter while vector or matrix scaling learns a linear transformation of the classifier outputs [4]. Later, additional regularization constraints such as penalizing off-diagonal terms [11] and order-preserving constraints [18] are introduced to improve matrix scaling. While several practical methods are developed in this regime, it was not clear previously whether learning a calibration function in a post-hoc manner would lead to calibration in the theoretical sense. We precisely answer this question and provide a theoretical justification of these methods. Even though, the proof is provided for negative-log loss, it is applicable to any proper loss function [2, 19].

We would like to clarify that our theoretical result (similar to [1]) is obtained under the assumption that the calibration set matches the true data distribution (or simply the test set distribution) which may not hold in practice. In this regard, there are various techniques such as label smoothing [14] and data augmentation [23] have been introduced to avoid overfitting while training the base network for calibration and we believe those techniques are applicable in our context as well.

6 Experiments

6.1 Experimental setup

For our experimental validation we use CIFAR-10 [10] and SVHN [16] datasets. For the base network f we use pre-trained models of different architectures (ResNet [6], ResNet Stochastic Depth [8], DenseNet [7], and Wide ResNet [22]). We train the proposed g-layers on a validation set (not used for training base networks) of CIFAR-10 and SVHN datasets, and evaluate the calibration error on the test set, unless specified otherwise. The g-layers are initialized with transparent layers so that at initialization they represent an identity mapping similar to the idea in [3]. The training of g-layers is then performed using SGD with momentum of 0.9, initial learning rate of 0.01 and ran for 20 epochs for CIFAR-10 and 50 epochs for SVHN. Furthermore, the learning rate is decayed by a factor of 0.1 at epochs 10, 20 and 30. To reduce overfitting of g-layers on the small calibration set, we use a weight decay of 0.001.

6.2 Calibration metric

Note that, as discussed in section 3, our theory as well as our approach guarantees multi-class-calibration. However, in the calibration literature [4, 11], the standard practice is to measure the calibration of top-1 predictions (or generally classwise-calibration). To this end, we measure the expected calibration error using Kolmogorov-Smirnov calibration error (KS-error) [5]. Note, ECE [15] is a widely used metric, however, since ECE relies on histograms (see fig 2 (a) and (b)) which is deemed as a weakness since the final error depends on the chosen histogram binning scheme. Furthermore, ECE is particularly unsuitable on deep networks trained on small datasets such as CIFAR-10, since over 90% of scores are over 0.9, and hence lie in a single bin (see fig 2 (c), which shows the plot of scores versus fractile). Therefore, we choose KS-error over ECE as KS uses a binning-free approach via cumulative distributions, which we briefly discuss below and refer the interested reader to [5, 9, 20].

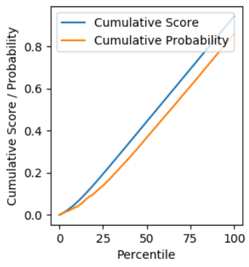
KS-error. Let (x_i, k_i) be samples, and let c_i be a chosen label for each sample i. To test the calibration of a specific class k, one chooses $c_i = k$ for all i. To test calibration on the top-1 class, $c_i = \mathrm{argmax}_k(h_k(x_i))$, where $h_k(x_i)$ denotes the classifier output for class k for the input sample x_i. In our case, $h = q \circ g \circ f$, where f is the base network, g is the added layers, and q denotes the softmax function. One can also test calibration for the second-top (or r-th top, or even within-top-r) choice of the network by defining c_i to be the label with the second-top (or r-th top) score. The KS-error computes the maximum difference between cumulative distributions as follows. For $t \in [0, 1]$ define

$$\sigma(t) = \frac{1}{M} \sum_{h_{c_i}(x_i) \le t} h_{c_i}(x_i) \ , \qquad \tau(t) = \frac{1}{M} \sum_{h_{c_i}(x_i) \le t} \mathbf{1}(k_i = c_i) \ , \tag{4}$$

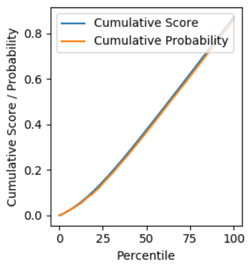
where M is the number of samples in the test set and $\mathbf{1}(\cdot)$ is 1 if its argument is true, and 0 otherwise. Then the metric is $\mathrm{KS} = \max_t |\sigma(t) - \tau(t)|$. Note, σ and τ are fast to compute using accumulated scores and accuracies from the network outputs sorted by $h_{c_i}(x_i)$. If the network is consistently over- or under-calibrated independent of the score $h_{c_i}(x_i)$, which is usually the case, then the KS-error measures the (empirical) expected absolute difference between score $h_{c_i}(x_i)$ and probability $P(c_i \mid h_{c_i}(x_i))$. This also provides visualizations similar to reliability diagrams.

6.3 Results

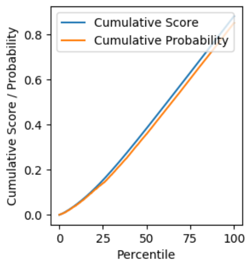
We first discuss a controlled experiment to understand the interplay between the number of parameters in g-layers and overfitting with respect to calibration for a given dataset. Later, we provide



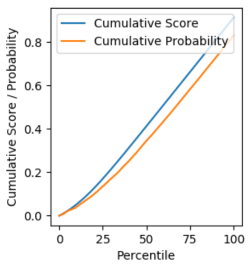
(a) *UC (KS: 8.3%)*



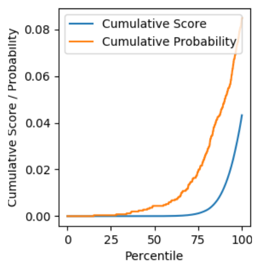
(b) *D1 (KS: 1.3%)*



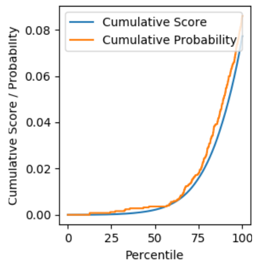
(c) *D2 (KS: 3.0%)*



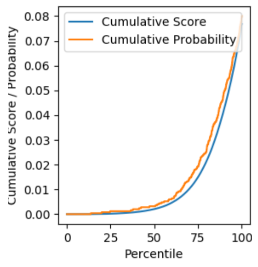
(d) *D3 (KS: 8.2%)*



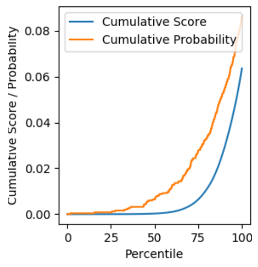
(e) *UC (KS: 4.2%)*



(f) *D1 (KS: 1.0%)*



(g) *D2 (KS: 0.8%)*



(h) *D3 (KS: 2.7%)*

Figure 1: *Visualisation of KS-Error on an unseen test set for uncalibrated model (a/e) and g-layers with 1–3 dense layers (b)–(d) and (f)–(h), for the top-ranked class (a)–(d) and the second highest-ranked class (e)–(h). The cumulative score (σ) and probability (τ) are plotted as functions of the data percentile. We observe that the g-layer models with 1 or 2 dense layers show (much) better calibration, the effect is more clear on the second highest class. However adding more layers will cause overfitting on the (small) calibration set and results in poorer calibration.*

experimental comparison with other post-hoc calibration methods. In short, as predicted by our theory, if overfitting to the calibration set is reduced in practice, learning complete g-layers lead to superior calibration. Nevertheless, heuristics for mitigating overfitting such as using larger calibration set and increased regularization (dropout, weight-decay, etc.) are relevant and the best approach to avoid overfitting with respect to calibration remains an open question.

Interplay between the number of g-layers and calibration. In the first set of experiments we use a relatively small ResNet-56, trained on CIFAR-10 and evaluate the effect of using 1, 2 or 3 g-layers with dimensionality 100 and ReLu activation functions. These additional layers are trained on 5,000 samples from a *calibration* set, and are evaluated on the remaining 5,000 samples of the test set (only for this experiment). We compare against the uncalibrated model for the top ranked class and the second-highest ranked class.

In fig 1, we plot $\sigma(t)$ and $\tau(t)$ to visualize the KS-error of the uncalibrated model (UC), and using 1–3 dense layers (D1 – D3). From the results we observe that the uncalibrated network gives poor calibration results and this is even clearer on the second-highest ranked class. For the top class (a)–(d) the scores show significant over-confidence in the result and by contrast, major under-confidence is observed for the second highest ranked class (e)–(g). In fact, the second-top class is far more likely to be the ground truth than is reflected by the score.

Much better calibration is obtained with 1 or 2 dense layers (D1 and D2). However, for 3 dense layers, performance on the test set is again relatively poor, whereas when evaluated on the calibration set used for training the calibration layers, the calibration is good – this is to be expected. Thus, having 3 calibration layers gives decreased performance, this is because of overfitting to the calibration set. This effect may be decreased if the calibration set is larger. Note adding 1 or 2 dense g-layers to a CNN, trained on a different set of samples than those used to train the base network, gives a large improvement in the calibration of the composite network, while having no deleterious effect on the accuracy. Training these g-layers adds about 1% to the training time of the network. For the remaining of the experiments we use g-layers with 2 dense layers.

Dataset	Model	Uncalibrated	Temp. Scaling	Vector Scaling	MS-ODIR	Dir-ODIR	g-**layer**
CIFAR-10	Resnet-110	4.750	0.916	0.996	0.977	1.060	**0.305**
	Resnet-110-SD	4.102	0.362	0.430	0.358	0.389	**0.279**
	DenseNet-40	5.493	0.900	0.890	0.897	1.057	**0.688**
	Wide Resnet-32	4.475	0.296	**0.267**	0.305	0.291	0.419
SVHN	Resnet-152-SD	0.852	0.552	0.570	0.573	0.607	**0.265**

Table 1: *KS calibration error [5] (in %) comparisons against state-of-the-art post-hoc calibration methods on image classification datasets: CIFAR-10 and SVHN, using various network architectures. Here, we use two dense g-layers with 32 units each for all our experiments.*

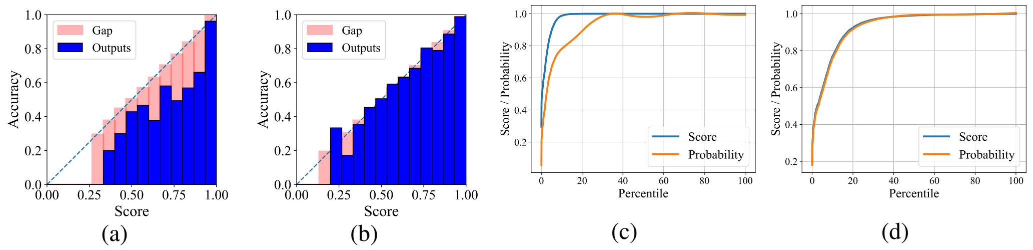


Figure 2: *Calibration graphs for an uncalibrated and calibrated ResNet-110 [7] trained on CIFAR-10 for the* top class *with a KS error of 4.8%, and top-*1 *accuracy of 93.6% on the unseen test set. Here, the network is calibrated using two g-layers with 32 dense units each. (**a**) and (**b**) show the reliability diagrams of scores versus probability for uncalibrated and calibrated network respectively. (**c**) and (**d**) show score and probability plotted against fractile for uncalibrated and calibrated network respectively. If the network is perfectly calibrated, the scores and probability plot will coincide with each other as can be observed in (**d**).*

Comparisons to other methods. In this set of experiments we compare g-layer calibration to several other calibration methods, including temperature scaling [4], vector scaling [4], MS-ODIR [11], and Dir-ODIR [11]. To provide the calibration results using the baseline methods, we use the implementation of [11]. On CIFAR-10, we compare different uncalibrated models (ResNet variants and DenseNet), and whereas on SVHN we evaluate a single ResNet-152-SD. Since, we do not need to retrain the base models, we train g-layers on top of the pretrained models.[4]

The performance is measured using KS-Error and the calibration results are presented in Table 1. Our method with just two g-layers layers with 32 units each consistently achieves comparable or even better calibration performance on all the tested case with negligible loss in accuracy ($< 0.5\%$) of the base network. To visually illustrate the efficacy of our method, we also show uncalibrated and calibrated graph (using standard reliability diagrams and plot of scores versus fractile) for our experiment on ResNet-110 trained on CIFAR-10 dataset in fig 2. It can be clearly observed that a simple two g-layers network, improves the calibration on an unseen test set substantially. According to our theory, the proposed method extends to large-scale datasets with a condition that a large calibration set needs to be used to train the additional g-layers to reduce overfitting.

We would like to point out that all the compared methods belong to the post-hoc calibration category and can be thought of as special cases of our method (that is learning a g-function). The main difference between these methods is the allowed function class while optimizing the g-layers, which can be thought of as a technique to avoid overfitting on a small calibration set.

7 Conclusion

The analysis in this paper gives broader conditions than previously known for a classifier such as a neural network to be correctly calibrated, ensuring that the network can be correctly calibrated during training, after early-stopping or through post-hoc calibration. This provides a theoretical basis for post-hoc calibration schemes. The method is validated by experiments with a simple extension to existing networks. We intend to study techniques to improve generalization with respect to calibration as a future work.

[4]Pre-trained models are obtained from `https://github.com/markus93/NN_calibration`.

References

[1] Christopher M Bishop. Mixture density networks. 1994.

[2] Andreas Buja, Werner Stuetzle, and Yi Shen. Loss functions for binary class probability estimation and classification: Structure and applications. *Technical report*, 2005.

[3] Tianqi Chen, Ian Goodfellow, and Jonathon Shlens. Net2net: Accelerating learning via knowledge transfer. *arXiv preprint arXiv:1511.05641*, 2015.

[4] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1321–1330. JMLR. org, 2017.

[5] Kartik Gupta, Amir Rahimi, Thalaiyasingam Ajanthan, Thomas Mensink, Cristian Sminchisescu, and Richard Hartley. Calibration of neural networks using splines. *CoRR*, 2020.

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

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

[8] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Weinberger. Deep networks with stochastic depth. In *European Conference on Computer Vision*, 2016.

[9] A Kolmogorov. Sulla determinazione empírica di uma legge di distribuzione. 1933.

[10] Alex Krizhevsky et al. Learning multiple layers of features from tiny images. 2009.

[11] 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. In *Advances in Neural Information Processing Systems*, pages 12295–12305, 2019.

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

[13] Ananya Kumar, Percy S Liang, and Tengyu Ma. Verified uncertainty calibration. In *Advances in Neural Information Processing Systems*, pages 3787–3798, 2019.

[14] Rafael Müller, Simon Kornblith, and Geoffrey E Hinton. When does label smoothing help? In *Advances in Neural Information Processing Systems*, pages 4696–4705, 2019.

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

[16] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.

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

[18] Amir Rahimi, Amirreza Shaban, Ching-An Cheng, Byron Boots, and Richard Hartley. Intra order-preserving functions for calibration of multi-class neural networks. *arXiv preprint arXiv:2003.06820*, 2020.

[19] Mark D Reid and Robert C Williamson. Composite binary losses. *Journal of Machine Learning Research*, 2010.

[20] Nikolai Smirnov. On the estimation of the discrepancy between empirical curves of distribution for two independent samples. 1939.

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

[22] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

[23] Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *6th International Conference on Learning Representations, ICLR 2018*, 2018.

Appendices

Here, we first provide the proofs of the results of our main paper and then provide additional results.

A Proofs of post-hoc calibration

A.1 A lemma

The following result will be useful. It is perhaps relatively obvious, but worth stating exactly.

Lemma A.6. *Let X be a random variable with values in $\mathcal{D}_X$ and $f : \mathcal{D}_X \to \mathcal{D}_Z$ be a measurable function. Let $h : \mathcal{D}_Z \to \mathbb{R}$ be a measurable function. If $Z = f(X)$, then*

$$E_{x\sim X}\, h(f(x)) = E_{z\sim Z}\, h(z) \ .$$

The proof follows from the definition of expected value, using a simple change of variables.

Proof. (sketch) The expectations may be written as

$$E_{x\sim X}\, h(f(x)) = \int h(f(x))\, d\mu_X$$
$$E_{z\sim Z}\, h(z) = \int h(z)\, d\mu_Z \ ,$$

where μ_X and μ_Z are probability measures on $\mathcal{D}_X$ and $\mathcal{D}_Z$. The desired equality then follows from a change of variables. □

The lemma can be given a more informal but more intuitive proof as follows. The expected value $E_{x\sim X}\, h(f(x))$ can be computed by sampling x from X and taking the mean of the values $h(f(x))$. In the limit as the number of samples increases, this mean converges to the $E_{x\sim X}\, h(f(x))$.

Similarly, $E_{z\sim Z}\, h(z)$ is obtained by sampling from Z and computing the mean of the values $h(z)$. However, random samples from Z are obtained by sampling $x \sim X$ for then $z = f(x)$ is a sample from the distribution Z. Hence, the two expectations give the same result.

A.2 Submersions

We are interested in submersions from $\mathbb{R}^m$ to $N = \Delta^{n-1}$, the standard open simplex.

Proposition A.7. *Let $q : \mathbb{R}^m \to \Delta^{n-1}$ be a submersion. If $\sum_{k=1}^n w_k\, \partial q_k/\partial z_j = 0$, then w_k is a constant for all k.*

Proof. Since Δ^{n-1} has dimension $n-1$, if q is a submersion, the Jacobian $\partial q_k(z)/\partial z_j$ has rank $n-1$. Since $\sum_{k-1}^n q_k(z) = 1$, taking derivatives gives $\sum_{k=1}^n \partial q_k(z)/\partial z_j = 0$ for all j. Written in terms of matrices, with $\mathtt{J} = \partial q_k(z)/\partial z_j$ this says that $\mathbf{1}^\top \mathtt{J} = 0$. Further, since $\mathtt{J}$ has rank $n-1$, if $\mathbf{w}^\top \mathtt{J} = 0$ then $\mathbf{w} = \alpha\mathbf{1}$.

A.3 Negative-logarithm loss and calibration

The following theorem is a basic known (in some form) property of the Negative-Logarithm cost function. The paper [1] gives the essential idea of the proof, but the theorem is not stated formally there.

Theorem A.8. *Consider joint random variables (Z, K), taking values in $\mathbb{R}^m$ and $\mathcal{K}$ respectively. Let $q : \mathbb{R}^m \to \Delta^{n-1}$ be a submersion. Define the loss*

$$L(q, Z, K) = -E_{(z,k)\sim(Z,K)} \log\big(q_k(z)\big) \ .$$

If

$$id = \underset{g:\mathbb{R}^m\to\mathbb{R}^m}{\operatorname{argmin}}\ L(q\circ g, Z, K)$$

then

$$P(k \mid z) = q_k(z) \ .$$

Proof. The assumption in this theorem is that the value of the loss function cannot be reduced by applying some function $g : \mathbb{R}^m \to \mathbb{R}^m$. We investigate what happens to the function $L(q, X, K)$ when q is replaced by the composition $q \circ g$, where $g : \mathbb{R}^m \to \mathbb{R}^m$ is some function. We compute

$$\begin{aligned} -L(q \circ g,\ Z, K) &= E_{(z,k)\sim(Z,K)} \log(q_k(g(z))) \\ &= \int \sum_{k=1}^{n} p(z,k) \log(q_k(g(z)))\, dz \end{aligned} \tag{5}$$

We wish to optimize (5) over $g : \mathbb{R}^m \to \mathbb{R}^m$. Consider optimization over a single dimension, j and let g' represent $\partial g/\partial z_j$. The Euler-Lagrange equation concerns a functional of the form $\int F(z, g, g')\, dz$, and says that the minimum (with respect to g) is attained when the Euler-Lagrange equation holds:

$$\frac{\partial F}{\partial g} = \frac{d}{dz}\frac{\partial F}{\partial g'}\ .$$

In this case, since g' does not appear in the functional, this gives $\partial F/\partial g = 0$. Since here $F(z, g, g') = \sum_{k=1}^{n} p(z,k) \log(q_k \circ g)$, we compute

$$\frac{\partial F}{\partial g} = \sum_{k=1}^{n} p(z,k) \frac{q_k' \circ g(z)}{q_k \circ g(z)} = 0\ .$$

Now, if this is optimized when $g = \mathrm{id}$, this becomes

$$\sum_{k=1}^{n} p(z,k) \frac{q_k'(z)}{q_k(z)} = 0\ . \tag{6}$$

If we define $L_k(z) = -\log(q_k(z))$, then (6) can be written as

$$\boxed{\sum_{k=1}^{n} p(z,k)\ \frac{\partial L_k(z)}{\partial z_j} = 0 \quad \text{for all } j\ .} \tag{7}$$

For the particular form of the negative-logarithm cost function given in the theorem, this becomes, for all j,

$$\begin{aligned} 0 &= \sum_{k=1}^{n} p(z,k)\ \frac{\partial}{\partial z_j}\Big(\log(q_k(z))\Big) \\ &= \sum_{k=1}^{n} \frac{p(z,k)}{q_k(z)}\ \ \frac{\partial q_k}{\partial z_j}\ . \end{aligned}$$

However, from Proposition A.7, this imples that $p(z,k)/q_k(z) = c$, a constant, so $p(z,k) = c\, q_k(z)$ for all k. However, since $\sum_k q_k(z) = 1$ and $\sum_k p(z,k) = p(z)$, this gives $p(z,k) = p(z)\, q_k(z)$, or $p(k \mid z) = q_k(z)$, as required. This completes the proof of Theorem A.8. □

A simple rewording of this theorem (changing the names of the variables) gives the following statement, which is essentially a formal statement of a result stated in [1].

Corollary A.9. *Consider joint random variables (X, K), taking values in $\mathcal{D}_X$ and $\mathcal{K}$ respectively, where $\mathcal{D}_X$ is some Cartesian space. Let $f : \mathcal{D}_X \to \Delta^{n-1}$ be a function. Define the loss*

$$L(f, X, K) = -E_{(x,k)\sim(X,K)} \log\left(f_k(x)\right)\ .$$

If

$$f = \underset{\hat{f}:\mathcal{D}_X \to \Delta^{n-1}}{\operatorname{argmin}}\ L(q \circ \hat{f}, X, K)$$

then

$$P(k \mid x) = f_k(x)\ .$$

This corollary follows directly from Theorem A.8 since if f minimizes the cost function over all functions, then it optimizes the cost over all functions $f \circ g$, where $g : \mathcal{D}_X \to \mathcal{D}_X$.

However, when $\mathcal{D}_X$ is a high-dimensional space (such as a space of images), then it may be a very difficult task to find the optimum function f exactly. Fortunately, a much less stringent condition is enough to ensure the conclusion of the theorem, and that the network (f) is calibrated.

Corollary A.10. *Consider joint random variables (X, K), taking values in $\mathcal{D}_X$ and $\mathcal{K}$ respectively. Let $f : \mathcal{D}_X \to \mathbb{R}^m$, and let $q : \mathbb{R}^m \to \Delta^{n-1}$ be a submersion. Define the loss*

$$L(q \circ f, X, K) = -E_{(x,k)\sim(X,K)} \log \big(q_k(f(x))\big) \ .$$

If f is is optimal with respect to recalibration, for this cost function, then $P(k \mid f(x)) = q_k(f(x))$.

Proof. Let $Z = f(X)$, and for any x define $f(x) = z$. Then, from lemma A.6,

$$\begin{aligned} L(q \circ f, X, K) &= -E_{(x,k)\sim(X,K)} \log \big(q_k(f(x))\big) \\ &= -E_{(z,k)\sim(Z,K)} \log \big(q_k(z)\big) \\ &= L(q, Z, K) \ . \end{aligned}$$

Then, according to Theorem A.8, if replacing q by $q \circ g$ will not decrease the value of the loss function (which is the definition that f is optimal with respect to recalibration), we may conclude that

$$P(k \mid z) = q_k(z) \ .$$

as required. □

B Multiclass and classwise calibration

We make the usual assumption of random variables X and K. Suppose that a function $f : \mathcal{D}_X \to \Delta^{n-1}$ is multiclass calibrated, which means that $P(k \mid z) = z_k$, where $z = f(x)$. We wish to show that it is classwise calibrated, meaning $P(k \mid z_k) = z_k$, and also that it is calibrated for top-r and within-top-r calibration. It was stated in [11] that classwise calibration, and calibration for the top class are "weaker" concepts of calibration, but no justification was given there. Hence, we fill that gap in the theorem below.

The proof is not altogether trivial, since certainly $P(k \mid z_k)$ is not equal to $P(k \mid z)$ in general. Neither does it follow from the fact that $Z = z$ implies $Z_k = z_k$.

First, we change notation just a little. Let $\hat{y}$ be the so-called 1-hot version of y, namely an indicator vector such that $\hat{y}_k = 1$ if $y = k$ and 0 otherwise. Then the condition for multi-class calibration is

$$P(\hat{y}_k = 1 \mid z_k = \sigma) = \sigma$$

The top-r prediction. We wish also to talk about calibration of the top-scoring class predictions. Suppose a classifier f is given with values in Δ^{n-1} and let $\hat{y}$ be the ground truth label. Let us use z^r to denote the r-th top score (so z^1 would denote the top score). Note that an upper index, such as in z^r here represents the r-th top value, whereas lower indices, such as z_k represent the k-th class. Similarly, define $\hat{y}^r$ to be 1 if the r-th top predicted class is the correct (ground-truth) choice, and 0 otherwise. The network is calibrated for the top-r predictor if for all scores σ,

$$P(\hat{y}^r = 1 \mid z^r = \sigma) = \sigma \ . \tag{8}$$

In words, the conditional probability that the top-r-th choice of the network is the correct choice, is equal to the r-th top score.

Similarly, one may consider probabilities that a datum belongs to one of the top-r scoring classes. The classifier is calibrated for being within-the-top-r classes if

$$P\big(\textstyle\sum_{s=1}^r \hat{y}^s = 1 \;\big|\; \sum_{s=1}^r z^s = \sigma\big) = \sigma \ . \tag{9}$$

Here, the sum on the left is 1 if the ground-truth label is among the top r choices, 0 otherwise, and the sum on the right is the sum of the top r scores.

Theorem B.11. *Suppose random variables X and K defined on $\mathcal{D}_X$ and $\mathcal{K}$ respectively, and let $f : \mathcal{D}_X \to \Delta^{n-1}$ be a measurable function. Suppose that $P(k \mid z) = z_k$, where $z = f(x)$. Then f is classwise calibrated, and also calibrated for top-r and within-top-r classes, as defined by (8) and (9).*

Proof. We assume that f is multiclass calibrated, so that $P(\hat{y}_k = 1 \mid z) = z_k$. First, we observe that

$$P(\hat{y}_k = 1, z) = P(\hat{y}_k = 1 \mid z)\, P(z) = z_k P(z)\ . \tag{10}$$

Then,

$$\begin{aligned} P(\hat{y}_k = 1 \mid z_k = \sigma) &= P(\hat{y}_k = 1, z_k = \sigma)\ /\ P(z_k = \sigma) \\ &= \int_{z_k=\sigma} P(\hat{y}_k = 1, z)\, dz\ /\ P(z_k = \sigma) \end{aligned}$$

where the integral marginalizes over all values of z with k-th entry equal to σ. Continuing, using (10) gives

$$\begin{aligned} P(\hat{y}_k = 1 \mid z_k = \sigma) &= \int_{z_k=\sigma} z_k P(z)\, dz\ /\ P(z_k = \sigma) \\ &= z_k \int_{z_k=\sigma} P(z)\, dz\ /\ P(z_k = \sigma) \\ &= z_k P(z_k = \sigma)\ /\ P(z_k = \sigma) = z_k \end{aligned}$$

which proves that f is classwise calibrated.

Next, we show that f is top-r calibrated. The proof is much the same, using top indices rather than lower indices. Analogously to (10), we have

$$P(\hat{y}^r = 1, z) = P(\hat{y}^r = 1 \mid z)\, P(z) = z^r P(z)\ . \tag{11}$$

This equation uses the equality $P(\hat{y}^r = 1 \mid z) = z^r$. To see this, fix z, and let k be the index of the r-th highest entry of z. Then $z^r = z_k$ and $\hat{y}^r = \hat{y}_k$. Then $P(\hat{y}^r = 1 \mid z) = P(\hat{y}_k = 1 \mid z) = z_k = z^r$.

Then,

$$\begin{aligned} P\big(\hat{y}^r = 1 \mid z^r = \sigma\big) &= P\big(\hat{y}^r = 1, z^r = \sigma\big)/P(z^r = \sigma) \\ &= \int_{z^r=\sigma} P\big(\hat{y}^r = 1, z\big)\, dz\ /\ P(z^r = \sigma) \\ &= z^r \int_{z^r=\sigma} P(z)\, dz\ /\ P(z^r = \sigma) \quad \text{from (11)} \\ &= z^r P(z^r = \sigma)\ /\ P(z^r = \sigma) \\ &= z^r\ . \end{aligned}$$

Here, the integral is over all z such that $z^r = \sigma$. This shows that f is top-r calibrated.

Finally, we prove within-top-r calibration. Refer to (9), let σ be fixed, and let z be some vector such that $\sum_{s=1}^r z^s = \sigma$. Since for $s = 1, \ldots, r$ the events $\hat{y}^s = 1$ are mutually exclusive, it follows that

$$P\big(\textstyle\sum_{s=1}^r \hat{y}^s = 1 \bigm| z\big) \ = \ \textstyle\sum_{s=1}^r P\big(\hat{y}^s \ = \ 1 \bigm| z\big) = \textstyle\sum_{s=1}^r z^s = \sigma\ .$$

This equality $P\big(\sum_{s=1}^r \hat{y}^s = 1 \bigm| z\big) = \sigma$ will hold for any z such that $\sum_{s=1}^r z^s = \sigma$. It follows that

$$P\big(\textstyle\sum_{s=1}^r \hat{y}^s = 1 \bigm| \textstyle\sum_{s=1}^r z^s = \sigma\big) = \sigma\ ,$$

as required.

Note the following justification for this last step. If some random variables A and Z satisfy $P(A = a \mid Z = z) = \sigma$ (a constant) for all z in some class C, then $P(A = a \mid z \in C) = \sigma$. For, the assumption implies that $P(A = a, Z = z) = \sigma P(Z = z)$. Now, integrating for $z \in C$ gives $P(A = a, Z \in C) = \sigma P(Z \in C)$, and hence $P(A = a \mid Z \in C) = \sigma$. $\square$

What this theorem is saying, for instance, is that the probability that the correct classification lies within the top 2 (or r) scoring classes, given that the sum of these two scores is σ, is equal to the sum of the two top scores.

The theorem can easily be extended to any set of classes, to show that if the classifier f is multiclass calibrated and S is any set of labels, that

$$P\big(\sum_{s\in S} \hat{y}^s = 1 \bigm| \sum_{s\in S} z^s = \sigma\big) = \sigma \ , \tag{12}$$

and

$$P\big(\sum_{k\in S} \hat{y}_k = 1 \bigm| \sum_{k\in S} z_k = \sigma\big) = \sigma \ . \tag{13}$$

C Additional Results

Here, we provide additional visualizations of calibration results using 1, 2 or 3 dense g-layers. For these experiments, similar to fig 1 in the main paper, we use ResNet-56 trained on CIFAR-10 dataset where the g-layers are trained on 5000 samples from the *calibration* set and evaluated on the remaining 5000 samples of the test set. The calibration results on the unseen test set for top-1, top-2, class-1, and class-2 predictions are shown in figures 3 – 6, respectively. Similar visualizations for the calibration set are given in figures 7 – 10. Note that, on calibration set, the results are unrealistically good and this is included to show how well the network minimizes the logarithmic loss, and hence achieves correct calibration on the dataset used to train the calibration. Evaluation on the "test" set shows how well the calibration generalizes to other images.

The titles of the graphs can be interpreted as follows. For instance, the graph `resnet56_cifar10_T182_Strain_Etest|class[-1]` means network ResNet-56, dataset CIFAR-10, trained for 182 epochs, on set "train", evaluated on set "test", results for class $[-1]$ (the top class). Additionally, `...._D1_T50_Scalib_Etest...` means the network has (in addition to the base network) 1 dense layer, trained for 50 epochs on set "calibration", evaluated on set "test".

In the title are also given the KS-error (measure of calibration) and Probability (the probability, or frequency that the selected class corresponds to ground truth). Thus for the top-class, "probability" means the accuracy of the network, how often it predicts ground truth. This is equal to 0.860 for the base network (top-left of fig 3). For the second-top class, "probability" represents the frequency with which the second-top class is the ground truth (0.085, as shown at the top-left of fig 4). For classes 1 and 2 "probability" shows the frequency with which the given class is the ground-truth. This is about 10%, because the test set contains 10 classes. In these visualizations, plots of cumulative scores and probabilities are shown in the top row of each graph, and scores and probabilities in the bottom row. They are plotted against percentile of the test set (left) and against the score (right).

Observations. Various observations about the results follow.

1. The uncalibrated (base) network gives poor calibration results, when tested on data (the "test" or "calibration" sets) not used for training. This is shown in the top-left graph in each set.
2. Much better calibration is obtained with 1 or 2 dense layers. However, for 3 dense layers, performance on the test set is again relatively poor, whereas when evaluated on the "calibration" set used for training the calibration layers, the calibration is good – this is to be expected. Thus, having 3 calibration layers gives decreased performance, because of over-fitting to the "calibration" set. This effect may be decreased if the "calibration" set is larger.
3. It is notable that for the top class (fig 3) the scores show significant over-confidence in the result. For the second-top class (fig 4) they show major under-confidence. In fact, the second-top class is far more likely to be the ground truth than is reflected by the score.
4. The method gives a major improvement in calibration, as evaluated visually, or by the KS-score. In addition, this occurs without any significant decrease in the accuracy of the network. For instance for the top class, the uncalibrated network gives accuracy (probability) of 0.860 on the test set, whereas with one dense layer the accuracy is also 0.860, for two layers it is 0.851, only $< 1\%$ drop is observed.
5. The time to train the dense calibration layers is about 1 second per epoch (5000 images per epoch) for 50 epochs. Training of the base network takes about 30 seconds per epoch of $50,000$ images, for 182 epochs, about 100 times as long.

In summary, the conclusions of the main paper hold. Specifically, addition of 1 or 2 dense layers to a CNN, trained on a different set of samples than those used to train the base network, gives a large improvement in the calibration of the network, while having no deleterious effect on the accuracy. Training of the additional calibration layers adds about 1% to the training time of the network.

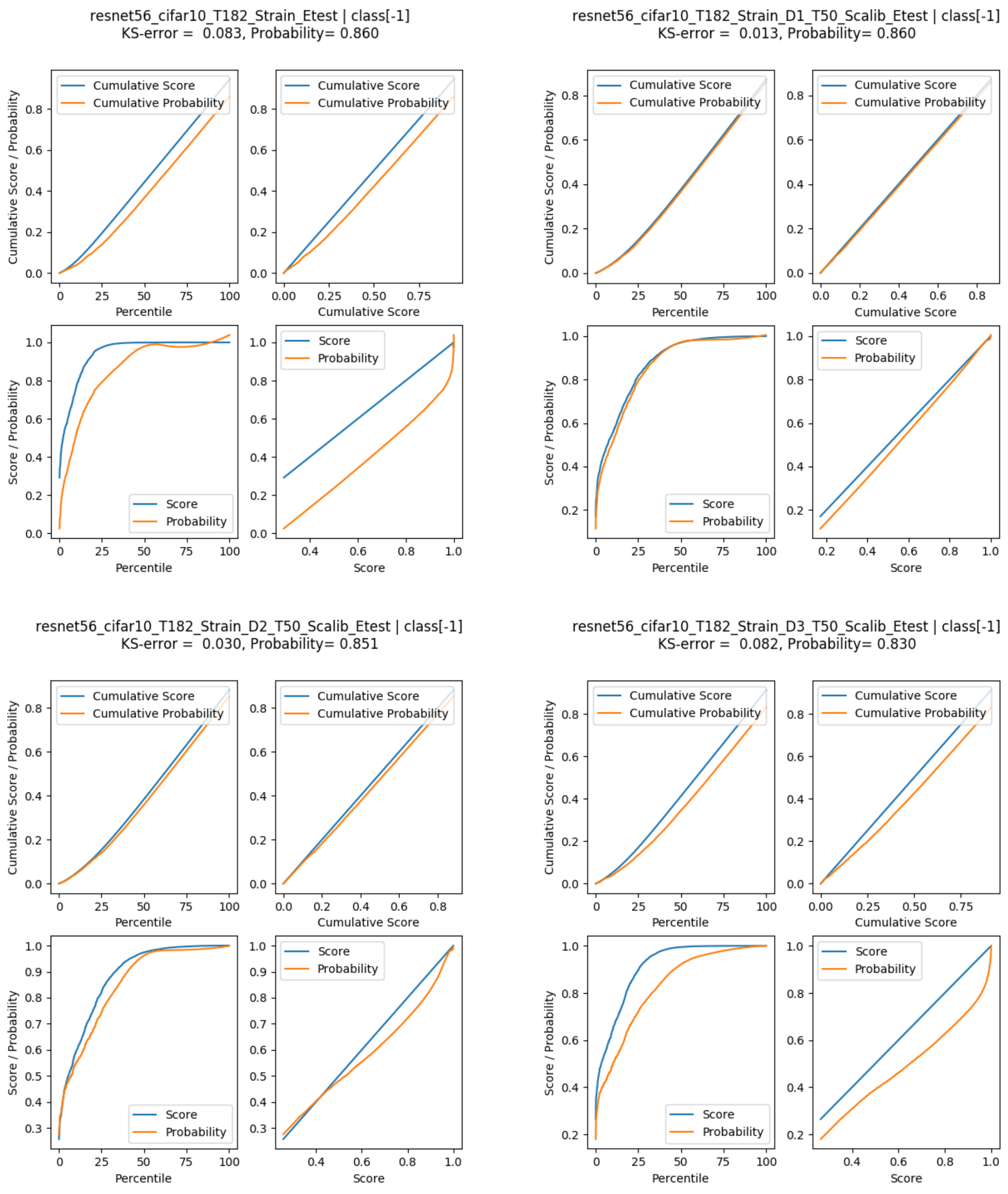


Figure 3: *Top class, uncalibrated and calibrated with* $0, \ldots, 3$ *dense calibration layers, evaluated on the test set.*

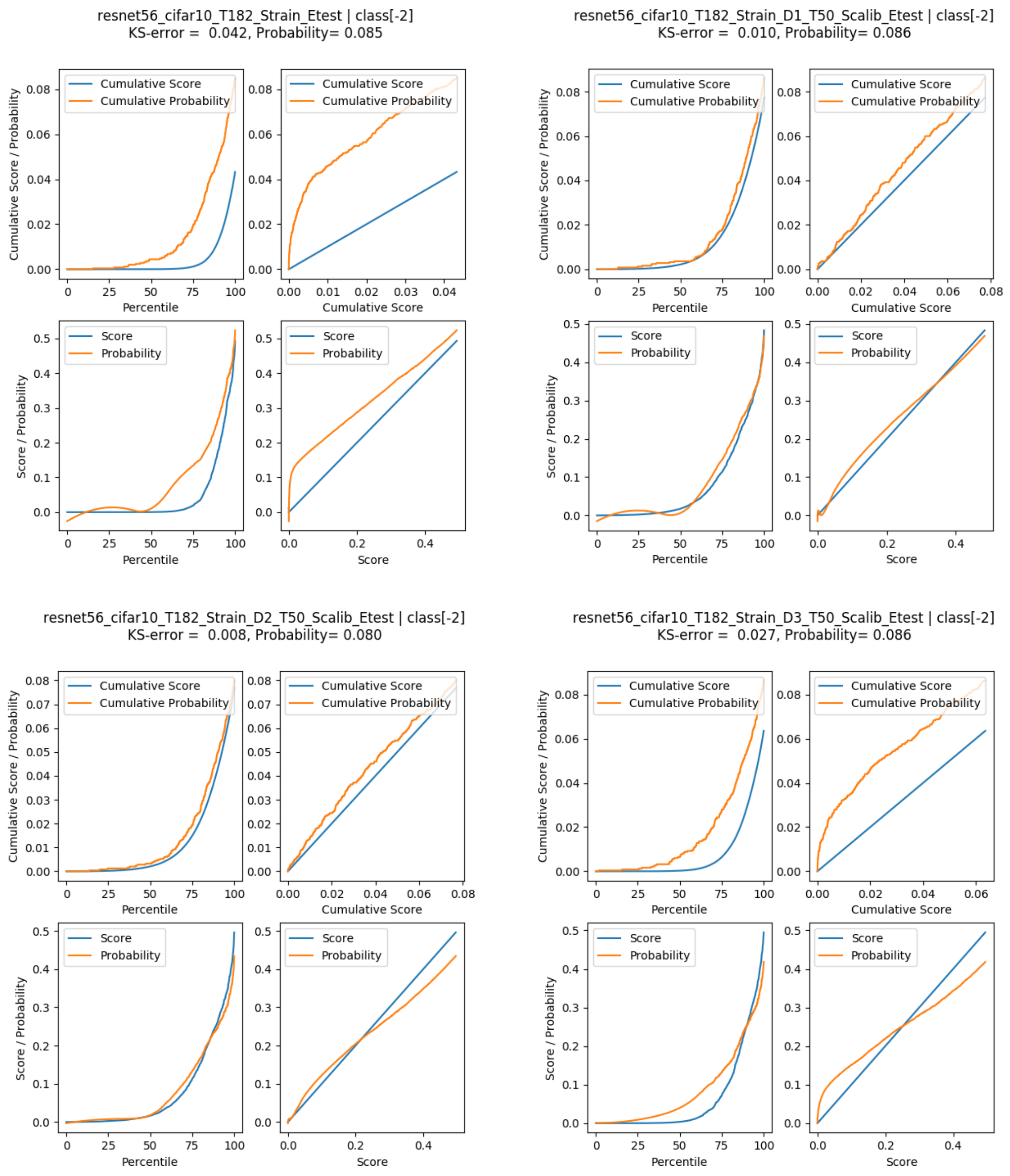


Figure 4: *Second top class, uncalibrated and calibrated with* $0, \ldots, 3$ *dense calibration layers, evaluated on the test set.*

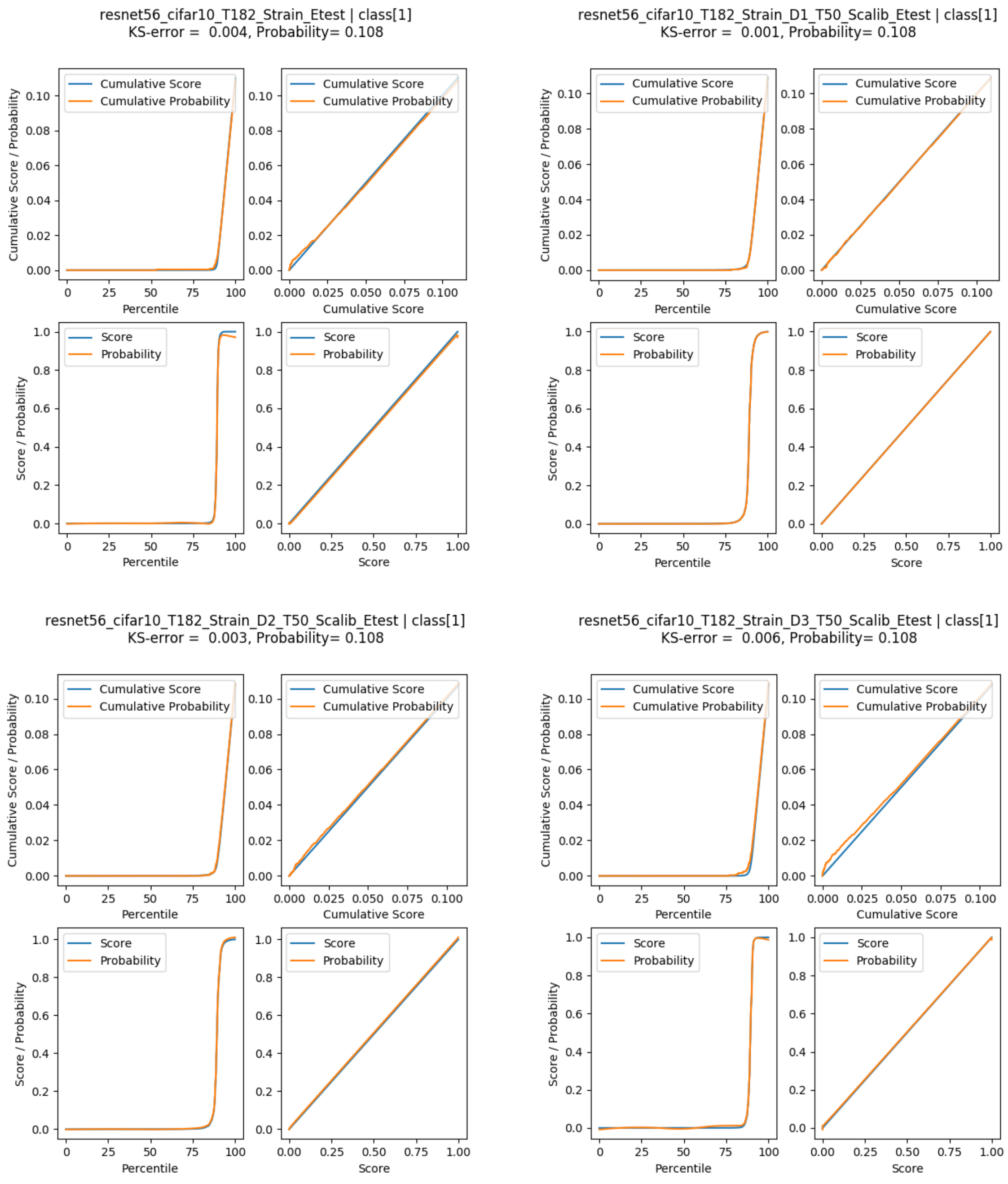


Figure 5: *Class* 1*, uncalibrated and calibrated with* $0, \ldots, 3$ *dense calibration layers, evaluated on the test set.*

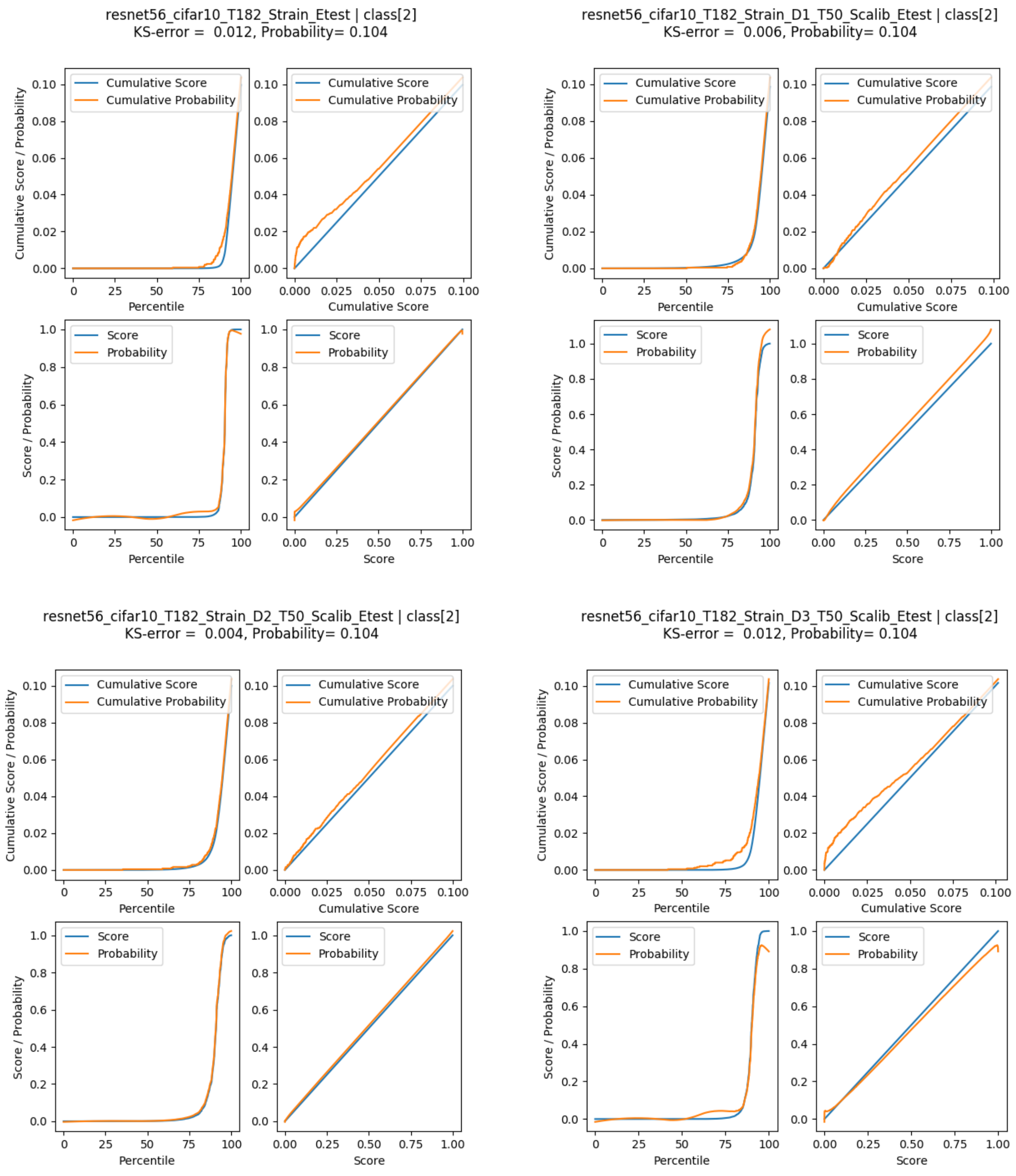


Figure 6: *Class* 2*, uncalibrated and calibrated with* $0, \ldots, 3$ *dense calibration layers, evaluated on the test set.*

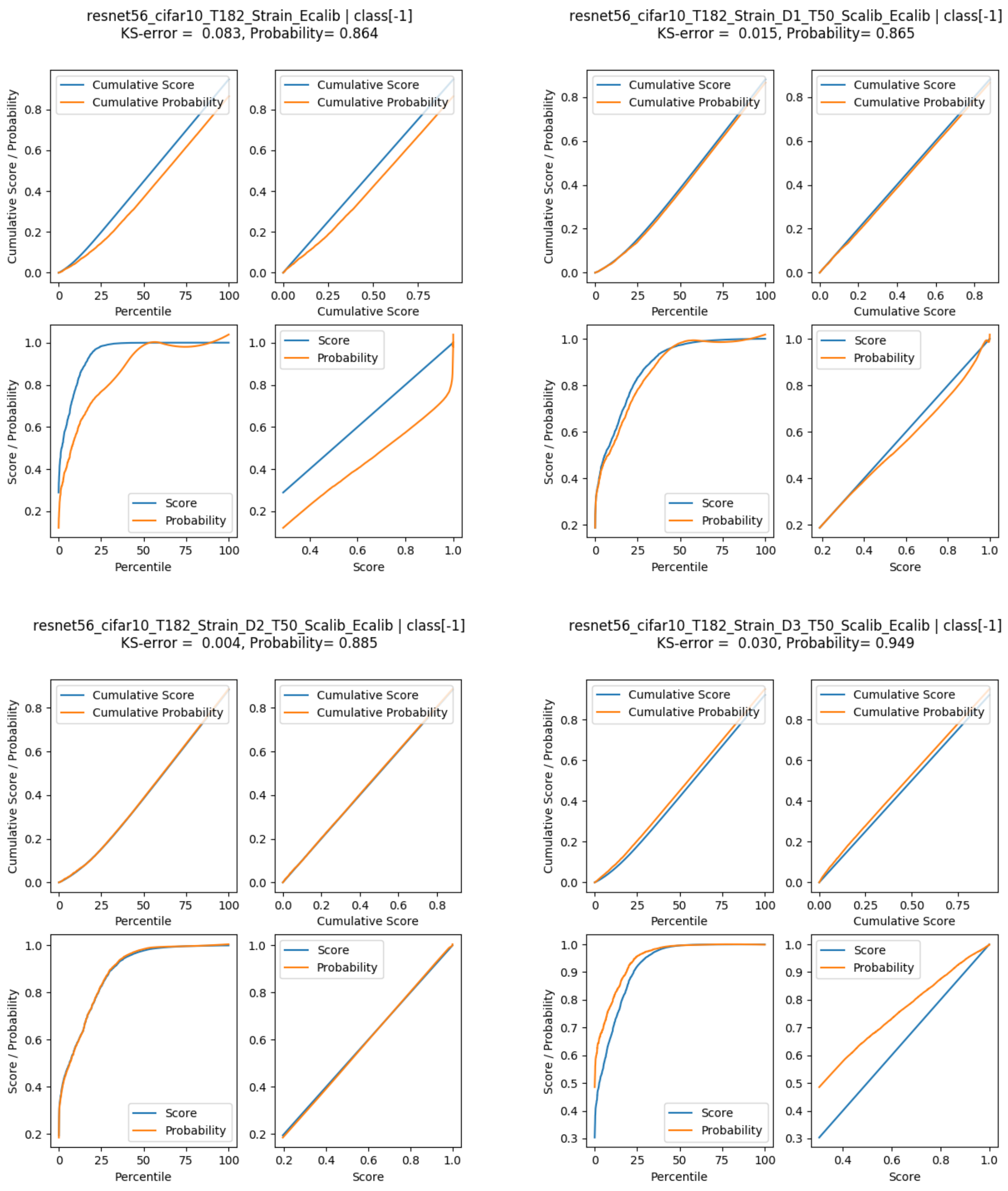


Figure 7: *Top class, uncalibrated and calibrated with* $0, \ldots, 3$ *dense calibration layers, evaluated on the calibration set.*

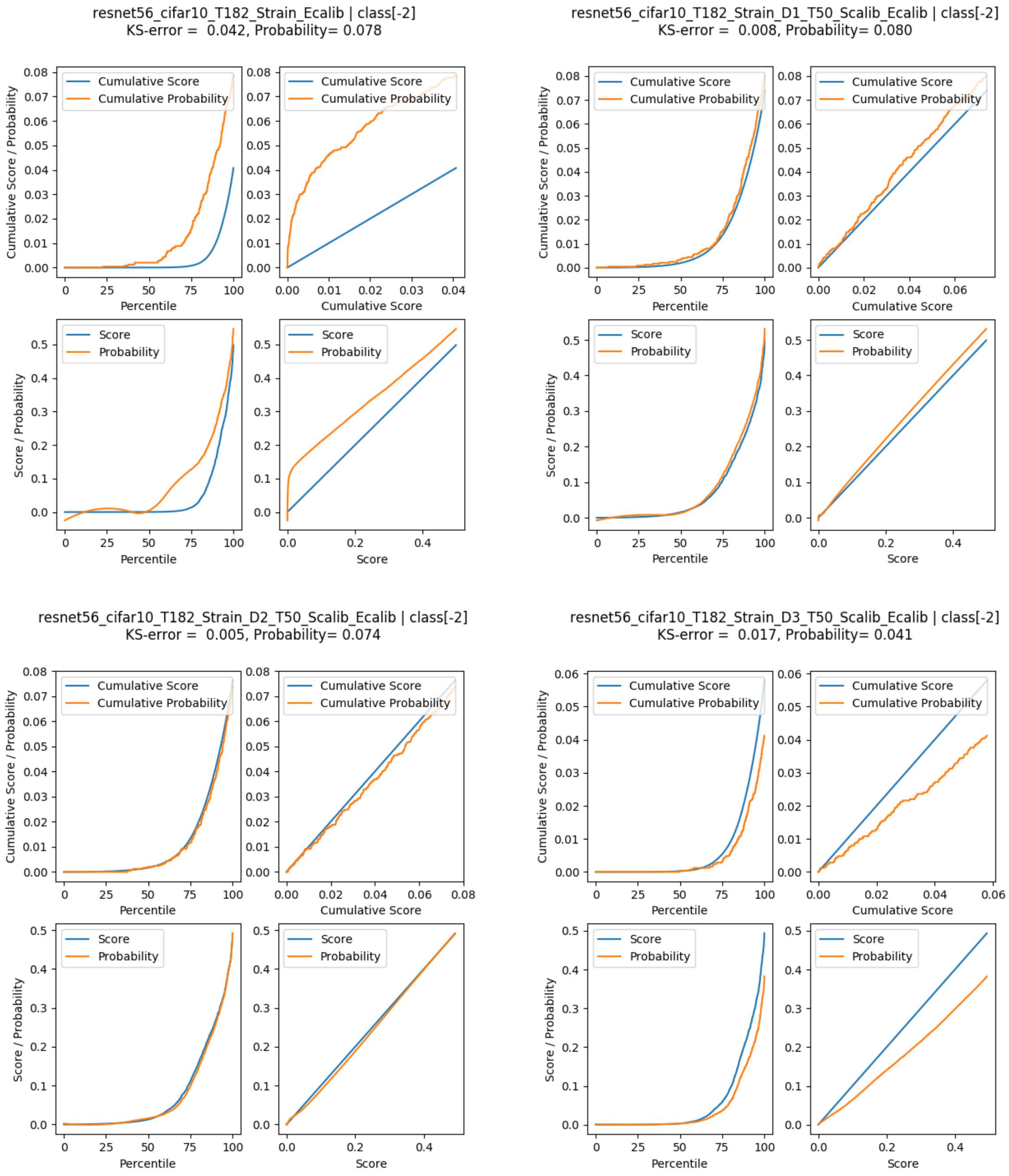


Figure 8: *Second top class, uncalibrated and calibrated with* $0, \ldots, 3$ *dense calibration layers, evaluated on the calibration set.*

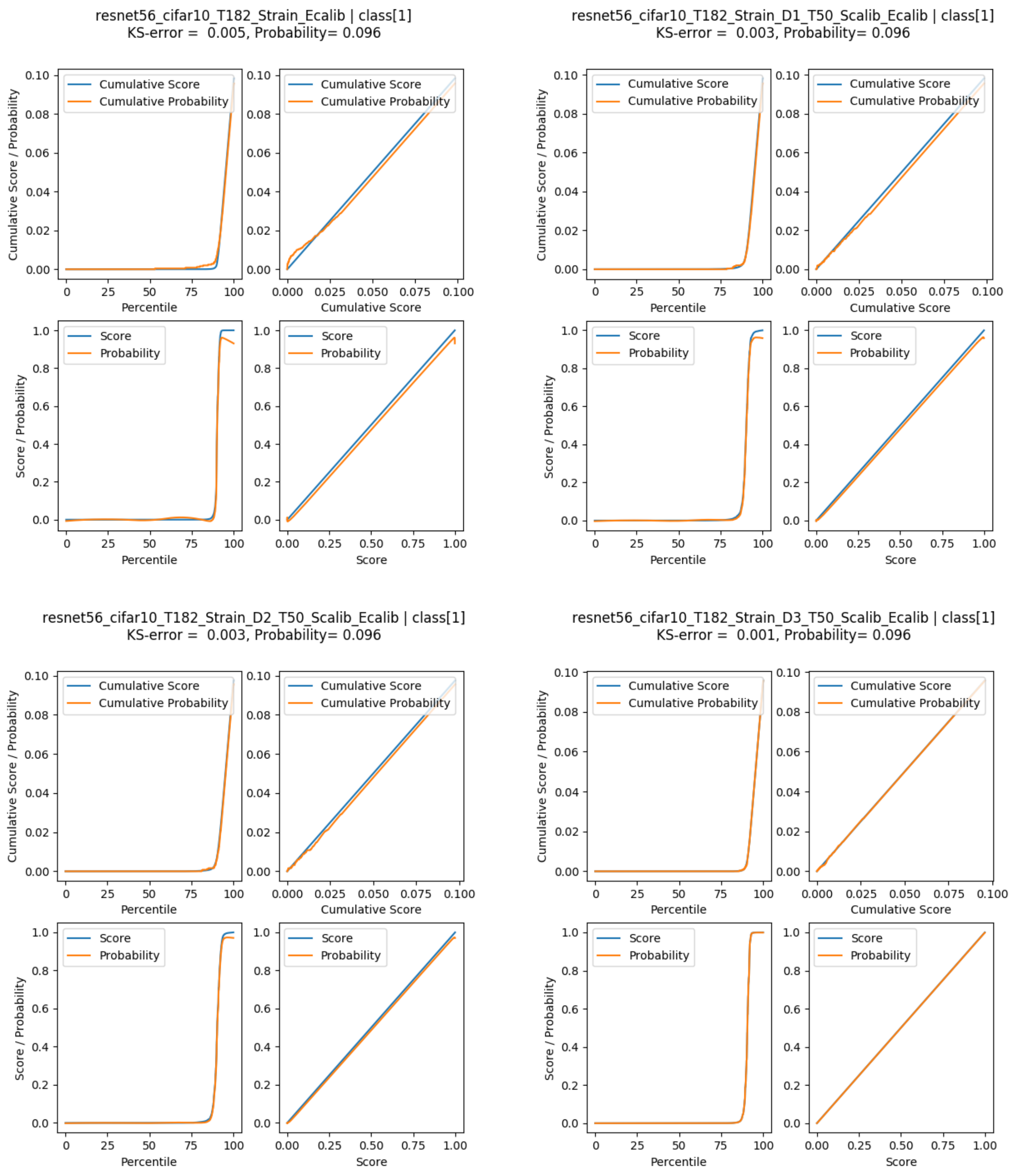


Figure 9: *Class* 1*, uncalibrated and calibrated with* $0, \ldots, 3$ *dense calibration layers, evaluated on the calibration set.*

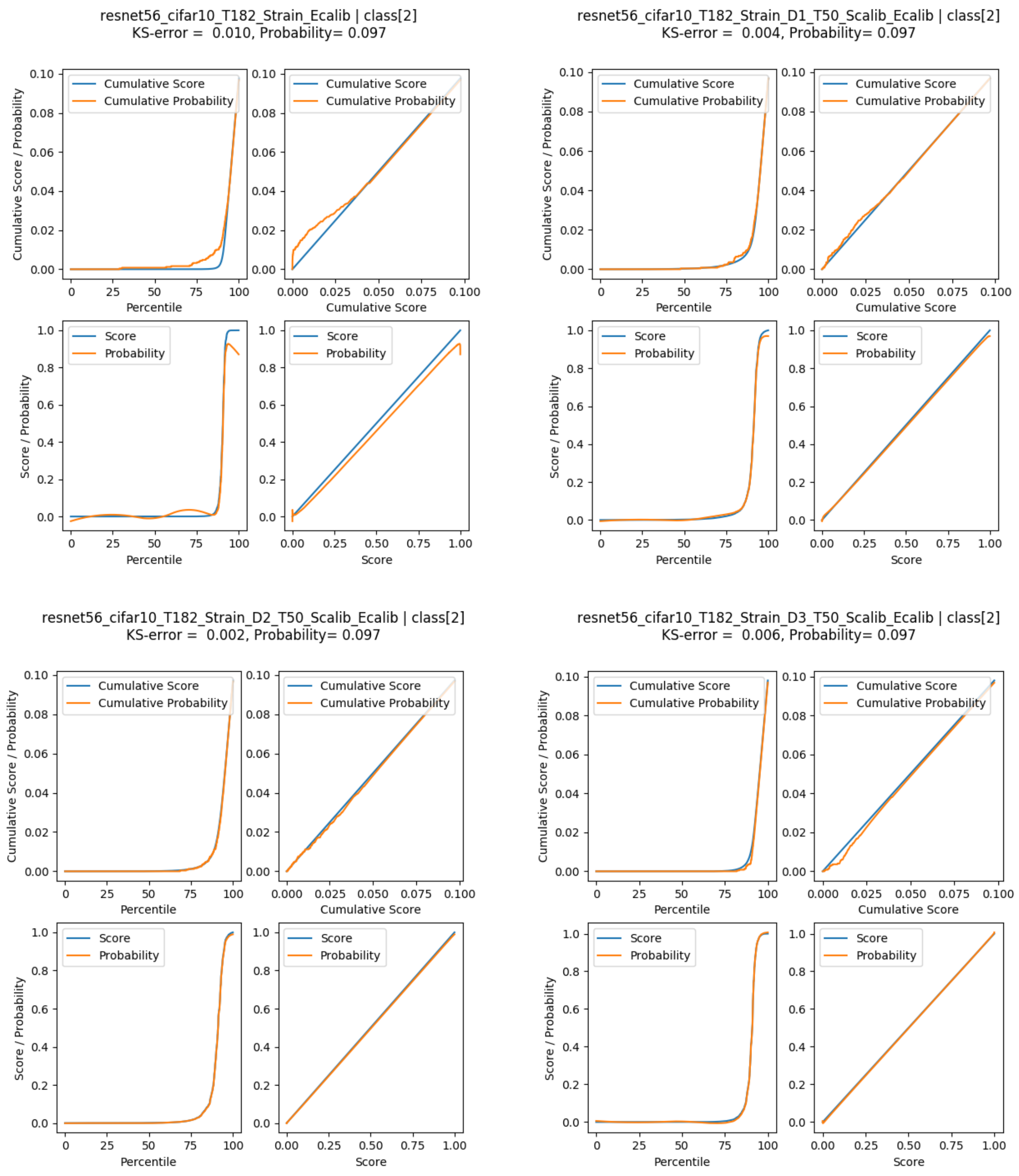


Figure 10: *Class* 2*, uncalibrated and calibrated with* $0, \ldots, 3$ *dense calibration layers, evaluated on the calibration set.*