Calibration of Classifiers

MICHAEL PANCHENKO, appliedAI

April 29th, 2021

We discuss the calibration of probabilistic classifiers, i.e. the question of whether the output vectors of such classifiers accurately approximate true class probabilities (or, in practical terms, frequencies on a test set). Despite being of relevance for many applications which need to model uncertainty and the fact that many commonly used neural networks tend to be miscalibrated, this topic is underrepresented in the ML community. We explain the subtleties involved in measuring calibration and recalibrating models.

Contents

1 An overview of calibration

In practical applications, output vectors of probabilistic classifiers such as neural networks or random forests are always used for decision making in one way or another. Depending on the context of the decision, certain properties of these output vectors might be more relevant than others. For example, if the task at hand is purely a classification - say an input image needs to be labeled as frog, cat or human by the model - only the prediction, i.e. the argmax of the output vector plays a role. A slightly more demanding application of the same classifier would be to label an image as “unknown” if the confidence in the prediction is below a certain threshold. Then, not only the argmax but also the actual value of the maximal confidence is needed. Even more demanding decisions can require the entire confidence vector, for example the decision whether to turn the wheel of a car at high speed to avoid hitting something, at the cost of risking an accident. In such a situation, it is important to take *all* confidences into account. We might want our system to behave differently when there is a 40% chance that the recognized object is a human vs. a 40% chance of it being a cat, even if the prediction is “frog” with 60% confidence in both cases

For decision making as in the last two examples, it is important that confidence vectors are meaningful quantities. Ideally, we would like them to correspond to real probabilities in some sense - this is why we call the classifier probabilistic after all. Unfortunately, modern neural networks trained with gradient descent (but also other models like random forests), despite often being very accurate, tend not to output empirically valid probabilities. This phenomenon is often called “miscalibration of modern neural networks”.

As we will explain in the sections below, calibration is a topic with many nuances. While there is quite some active research, it is somehow underrepresented in the machine learning community. We believe that this is firstly because there are many situations in which only metrics related to accuracy matter, and secondly because even in situations in which calibration is very important, it is not entirely clear how to measure its effect. To remedy this and to bring measurements of calibration closer to the needs of practitioners, we propose an application-centric scheme for studying the effects of miscalibration in Sections 4 and 6. We also publish an open-source python library aimed at helping with all the topics related to calibration mentioned in this review, see [PSB].

1.1 Definition of strong calibration

A probabilistic classifier learns to predict confidences from inputs. For an input x, the output of the classifier is a **confidence vector** which we denote $c(x)=(c_1(x),\ldots,c_K(x))$, where K is the number of classes. For our purposes, we can assume that c is a deterministic function of x. In most of what follows, we will not be concerned with the input, so we will omit the dependency on it. This means that c will typically denote the confidence vectors themselves and, sometimes, the classifier as a function mapping inputs to confidences; it should always be clear from the context what is meant by c. We assume that the confidences are always properly normalized, so that $\sum c_k = 1$.

Loosely speaking, we call a classifier *calibrated* if at test time the confidence vectors represent true probabilities, i.e. when for all $k \in \{1,\ldots,K\}$:

$$p_{\text{observed}}(\text{true label is } k \mid \text{confidence is } c) = c_k.$$

We can formulate this more rigorously as follows: Let X, Y be random variables representing inputs and ground truth labels. The random variable $C := c(X)$ of confidence vectors takes values in the $(K-1)$-dimensional simplex Δ^{K-1}. We are interested in the true probabilities of the ground truth labels given that a certain confidence has been observed, i.e. in the distribution of $Y \mid C$, which we will approximate using a test set. The vector of quantities $\mathbb{P}(Y=k|C)$, $k \in \{1\ldots K\}$ is a deterministic function of C: the **canonical calibration function**, r: $\Delta^{K-1} \longrightarrow \Delta^{K-1}$, given by[1]

$$r(C) := \mathbb{P}(Y \mid C). \tag{1}$$

When the distributions have densities, $r_k(c) = p(k \mid c)$, where $p(k|c)$ is the conditional density of $Y|C$. Because we will take expectations wrt. to C, we are only interested in the region of Δ^{K-1} where $p(c) > 0$. Assuming that (Y, C) has a density, the canonical calibration function is well defined everywhere in this set [Kle, Ex. 8.31]. For this reason,

[1] Here, and from now on, we use vector notation for class probabilities: when we write Y we mean $Y=k$ for all k

in the sequel we will assume in computations that $p(y|c) = 0$ where $p(c) = 0$.

Note that, because C is a random variable, $r \circ C$ is also a random variable. A probabilistic classifier c is called **strongly calibrated** iff the canonical calibration function is the identity, i.e. iff

$$c(X) = r(c(X)). \tag{2}$$

Any metric that vanishes when C and $r \circ C$ are equal almost surely, or equivalently when r is the identity function almost everywhere, is thus a metric of strong calibration. In particular, the L_p norms on the simplex (the range of $c(X)$) form such metrics, and we have the condition that c is strongly calibrated iff

$$\|\mathrm{id} - r\|_p = 0.$$

In general, accuracy and calibration are, to an extent, independent of each other.[2] A classifier can be perfectly accurate but not calibrated and perfectly calibrated but inaccurate. As an example for the former, think of a classifier that always predicts correctly but does so with a confidence of $1/K + \varepsilon$ (where ε is some small positive number). The accuracy is 1 but clearly the classifier is under-confident, because being always correct means that $\mathbb{P}(Y = k|c(x)) = 1 \neq c_k(c) = 1/K + \varepsilon$, and hence it is not calibrated. On the other side of the spectrum, a classifier which always predicts the (constant) population fractions is perfectly calibrated but generally inaccurate.

1.2 Why models are miscalibrated

Usually the objective on which a probabilistic classifier is trained is a so-called **proper scoring rule** [LPB], which for our purposes we can interpret to mean that the training objective reaches its optimum iff the confidence vectors outputted by the classifier are equal to the true conditional distribution of labels given inputs. Cross entropy and the so-called Brier score (the usual MSE between predictions and one-hot encoded labels) are proper scoring rules [LPB]. If the predictive distribution of a classifier parametrized by θ is denoted as $p_\theta(y|x)$, then a proper scoring rule is optimized at θ^* iff[3]

$$p_{\theta^*}(y|x) = \mathbb{P}(Y = y|x). \tag{3}$$

This means that if a classifier optimizes a proper scoring rule, it is *both maximally accurate and perfectly calibrated*. In this case, we say that p_θ is the **best possible classifier**.

[2] Although we will show that in many situations of practical relevance there are general relations between both

[3] Once the classifier is fixed, the predictive probability $p_\theta(y|x)$ is the same as $c(x)$ in the previous section, i.e. $p_\theta(k|x) = c_k(x)$

Why, then, are modern neural networks often miscalibrated despite being trained on a proper scoring rule? The empirical observation is that they sacrifice calibration for accuracy, meaning that after a certain amount of training, the out-of-sample calibration begins to decline while the out-of-sample accuracy continues to improve. This was first noticed in [GPSW] where several aspects of neural network miscalibration are explained in detail. There is however no solid theoretical understanding of why this happens, although experimentally, higher model capacity, batch normalization as well as low or no weight decay tend to reduce calibration.

Since the scoring rules measure calibration and accuracy simultaneously, miscalibration cannot be reliably captured by purely evaluating the training and validation losses (although in experiments one does see a decrease in calibration manifest itself as an increase in the test loss) and one has to resort to metrics that specifically target calibration.

1.3 Strong calibration and best possible post-processing

The optimality condition of a proper scoring rule is very similar to the condition for being strongly calibrated. Rewriting equation (3) in terms of random variables we get: a classifier c optimizes a proper scoring rule iff (remember that we use vector notation for probabilities)

$$c(X) = \mathbb{P}(Y|X). \tag{4}$$

The only difference with respect to the definition of strong calibration is that we condition on X instead of $c(X)$ on the right hand side. This is not a coincidence: Imagine that instead of samples from X we observe samples from $g(X)$, for some g. Now we need a classifier taking $g(X)$ as input. The best possible classifier we could have, let us call it $c^\star|_g$, would optimize some proper scoring rule and fulfill

$$c^\star|_g\,(g(X)) = \mathbb{P}(Y|\,g(X)).$$

Setting now $g = c$, the original classifier, follows that reaching strong calibration for the original problem just means obtaining *the best possible classifier given that one only observes* $c(X)$ *and not* X *directly*. We call this the **best possible post-processing classifier** and it fulfills:

$$c^\star|_c\,(c(X)) = \mathbb{P}(Y|c(X)) = r(c(X)),$$

where r is the canonical calibration function for c. We have then

$$c^\star|_c = r,$$

and the immediate observation that *any classifier c that is already strongly calibrated cannot be improved through post-processing*, where “improvement” means according to some metric of strong calibration, e.g. the L_p norms mentioned above.

One property differentiating the analysis of calibration from classification after generic preprocessing is that by definition the inputs $c(X)$ to the function r will live in the same space as the outputs, namely in Δ^{K-1}. Therefore, the canonical calibration function and approximations to it can be interpreted geometrically as simplex automorphisms.

Thus, optimizing calibration by post-processing the classifier's output is equivalent to finding the overall best possible post-processing classifier, and the optimum is an approximation to the canonical calibration function r. One way of obtaining it is to fit a parametrized function using a proper scoring rule, which will often be the same loss used to train the original classifier.[4] The choice of this function is important, for it should counteract the tendency to over-fit in favor of accuracy described above. Stacking a neural network with the same tendency on top of another neural network is not helpful (although stacking a small NN on a random forest which was not trained on a proper score might work). Therefore, will need to resort to more restricted families of functions. The simplest example considers constant multiplications of the logits. Because these do not affect accuracy, a decrease in the post-processing training loss will be a reliable indicator of improved calibration. See Section 5.1 for details.

1.4 Other notions of calibration

Strong calibration is not the only useful calibration property one can define, although it is the most general one. Other notions of calibration can arise by conditioning on a function of C instead of on C itself. For example, by conditioning on the confidence in the predicted label, in a similar fashion as for the canonical calibration function (1) one can define the **confidence calibration function** $r_{\max}: [1/K, 1] \longrightarrow [0, 1]$ as

$$r_{\max}(\max(C)) := \mathbb{P}(Y = \operatorname{argmax}(C) | \max(C)).$$

As for strong calibration, one is only interested in $r_{\max}$ where it can be defined, i.e. where the density of $\max(C)$ is positive, and it is set to 0 everywhere else.

By conditioning on the confidence of a fixed label k, one arrives at the **class-wise calibration function** $r_k: [0, 1] \longrightarrow [0, 1]$, with

$$r_k(C_k) := \mathbb{P}(Y = k | C_k).$$

[4] Alternative non-parametric methods estimate r directly from the sample statistics. These tend to perform worse for large K, because of the difficulties involved in estimating probability distributions in high dimensions. For instance, the number of bins in typical binning schemes grows exponentially with the dimension

The classifier is called **confidence calibrated** iff $r_{\max} = \mathrm{id}$ and **classwise calibrated** for class k iff $r_k = \mathrm{id}$. Below we define calibration metrics inspired by these reduced notions. Both definitions are special cases of the concept of a *calibration lens* introduced in [VWA+] which can be used to create calibration functions in a similar fashion. All such calibration functions are equal to the identity function if the strictly stronger condition of strong calibration holds.

The confidence and classwise calibration functions introduced above can be understood as canonical calibration functions of *induced binary classifiers* that predict two-dimensional confidence vectors as $(\max(C), 1 - \max(C))$ and $(C_k, 1 - C_k)$ respectively.[5] From the discussion in Section 1.3 one can conclude that a confidence- or class-wise calibrated classifier is the *best possible classifier that can be obtained by post-processing these induced classifiers.*

2 Measuring calibration

Unlike metrics focused on the quality of predictions like accuracy, precision, or recall, there is no consensus about the best way to measure (or visualize) calibration. There are several reasons for that.

1. **Definition:** $r(C)$ is a vector-valued quantity, as is $r(C) - C$. Depending on the situation at hand, different aspects of it are more relevant than others.

2. **Estimation:** Typically, expressions involving $r(C)$, like expectation values, are difficult to estimate using a finite set. This is especially true for problems where the number of classes K is large. As explained in Section 1.3, a good estimate of the canonical calibration function can be used to recalibrate the classifier, thus *estimating r directly is as hard as recalibrating*.

3. **Optimality:** While the commonly used losses are usually proper scoring rules and decreases in them over a test set can signalize better calibration, they generally do not vanish for strongly calibrated classifiers (nor do they vanish for the overall optimal classifier). Therefore, they are not proper measures of calibration.

4. **Comparison:** Numerical values of miscalibration should be comparable across different data distributions, something that does not happen e.g. for loss functions. Unfortunately, most common calibration metrics, like those introduced below, do not have this property.

Finally, while a useful classifier always has to be accurate to some degree, calibration is often optional or even unnecessary. E.g. for applications like picking the most probable class or clustering that only require the predicted value (the argmax), there might be no need to

[5] The induced labels are then defined as $Y_{\text{binary}} = \mathbb{1}_{\{\text{argmax}(C)\}}(Y)$ or $Y_{\text{cw}} = \mathbb{1}_{\{k\}}(Y)$ respectively

measure calibration at all.[6] This shows that in practical situations, measuring calibration should be done in a way that is useful for the problem at hand. We will elaborate on this in Section 4.

2.1 Calibration metrics and reliability curves

To overcome the difficulties outlined above, a scalar value representing miscalibration which vanishes when a classifier is strongly calibrated and an estimator for it.[7] The latter is often defined through binning. Such scalar values are called **calibration metrics**. They necessarily obscure some properties of $r(C)$ that might be relevant for applications. Thus, in a decision making scenario it might be necessary to consider multiple calibration metrics or to create a custom one.

The simplest metrics are obtained focusing on the calibration either of a single class, or of the predictions. They also give rise to visualizations of miscalibration, as we explain below. The well known **expected calibration error**, ECE, stems from confidence calibration and is defined as follows. First introduce the one dimensional random variable $Z := \max(C)$, which we assume to have density $p(z)$. The ECE is computed by conditioning on Z (as opposed to conditioning on the full confidence vector C as above) as[8]

$$\begin{aligned}\mathrm{ECE} &:= \mathbb{E}_Z\left(|\mathbb{P}(Y = \operatorname{argmax}(C) | Z) - Z|\right) \\ &= \int_{1/K}^{1} p(z) |\mathbb{P}(Y = \operatorname{argmax}(C) | Z = z) - z| \, \mathrm{d}z.\end{aligned}$$

The last equation also suggests a method to estimate the ECE by binning the one-dimensional domain of Z and evaluating the empirical probabilities within the bins. The integral then turns into a sum over the bins.

Note that the ECE is zero only if $|\mathbb{P}(Y = \operatorname{argmax}(C) | Z = z) - z|$ vanishes a.s. wrt. Z. This function can be conveniently visualized by plotting z on one axis and $\mathbb{P}(Y = \operatorname{argmax}(C) | Z = z)$ on the other. The resulting graph is called **reliability curve**. For a confidence calibrated classifier (and hence for a strongly calibrated one too), the reliability curve is exactly diagonal. Note that such a visualization can be misleading since it does not take into account $p(z)$, which for a binning scheme is approximated by the fractions of data points per bin and can be thought of as weights. The reliability curve might be far from the diagonal in some bins but the ECE could still be small if the population fractions in these bins are low. The intuition here is that the reliability curve just shows how miscalibrated a prediction is for all possible confidences while the ECE also takes into account how often such confidences are actually returned by the classifier.

Similarly, instead of focusing on the predictions we can calculate class-wise calibration errors and reliability curves for selected classes. Let $p(c_k)$ stand for the marginal probability density of the 1-dim.

[6] Nevertheless, one could argue the advantage of taking calibration into account for most decision problems using probabilistic classifiers

[7] However, we will see examples which vanish without the classifier being strongly calibrated

[8] For the domain of integration, note that the maximum of a K-dimensional confidence vector is always larger or equal than $1/K$, so $p(z) = 0$ for $z \notin [1/K, 1]$

random variable $C_k = c_k(X)$ - the confidence of class k.[9] We can define the **class-wise expected calibration error** cwECE_k as

$$\begin{aligned}\mathrm{cwECE}_k &:= \mathbb{E}_{C_k}(|\mathbb{P}(Y=k|\,C_k) - C_k|)\\ &= \int_0^1 p(c_k)|p(k|\,c_k) - c_k|\,\mathrm{d}c_k,\\ \mathrm{cwECE} &:= \frac{1}{K}\sum_k \mathrm{cwECE}_k.\end{aligned}$$

Again, one can estimate this with binning and visualize $|\mathbb{P}(Y=k|\,C_k = c_k) - c_k|$ using reliability curves. These approximate the reduced calibration functions defined in Section 1.4.

Note that conditioning on the prediction Z or a single confidence C_k is *not the same* as conditioning on the full vector, even after taking the expectation values. In other words, in general (note that the expectations are taken wrt. C):

$$\begin{aligned}\mathrm{cwECE}_k &\neq \mathbb{E}_C(|r_k(C) - C_k|),\\ \mathrm{ECE} &\neq \mathbb{E}_C(|P(Y = \mathrm{argmax}(C)|\,C) - \max(C)|).\end{aligned}$$

In fact, a classifier might have all class-wise calibration errors equal to zero (and also ECE being zero) but still not be strongly calibrated! See [VWA+] for an example. There has been effort to design metrics which are better at capturing strong calibration and to devise statistical tests for strong calibration [WLZ, VWA+]. However, demanding strong calibration might be too much to ask for specific applications, where only certain aspects of calibration may matter.

The methods outlined above demonstrate that there is no single one-fits-all solution for measuring calibration. The commonly used metrics always obscure some aspect of calibration and the full strong calibration is difficult to evaluate and may be not needed.

3 Computing and visualizing miscalibration

We now elaborate on the computation of calibration metrics and reliability curves using a finite data set D. We will focus on the ECE but the same considerations apply to cwECE and similar metrics.

3.1 Estimating naively with binning

The most straightforward way of estimating the ECE is with a constant binning scheme: Let $P := \left\{b_0 = \frac{1}{K}, \ldots, b_N\right\}$ be a uniform partition of the interval $[1/K, 1]$ in N bins $I_j = [b_{j-1}, b_j)$, with $I_N = [b_{N-1}, b_N]$. The corresponding estimator $\widehat{\mathrm{ECE}}_{P;D}$ takes the form

$$\widehat{\mathrm{ECE}}_{P;D} = \sum_{m=1}^{N} \frac{N_m}{|D|}|\mathrm{acc}_{m;D} - \mathrm{conf}_{m;D}|,$$

[9] Here, contrary to the density for the predicted class, $p(c_k)$ is defined on the whole interval $[0,1]$

where N_m is the number of samples in bin m, and, letting $Z = \max(C)$ as above,

$$\mathrm{acc}_{m;D} := \hat{\mathbb{P}}_D(Y = \mathrm{argmax}(C) \mid Z \in I_m),$$

$$\mathrm{conf}_{m;D} := \hat{\mathbb{E}}_D(Z \mid Z \in I_m) \approx \frac{b_m + b_{m+1}}{2},$$

are the empirical accuracy and average confidence in the bins respectively, and $\hat{\mathbb{P}}_D, \hat{\mathbb{E}}_D$ stand for the usual (counting) sample estimators of $\mathbb{P}, \mathbb{E}$ over the set D. We explicitly highlight the dependence on the dataset D in all relevant quantities.

With such a binning scheme we can immediately create a reliability curve by plotting the accuracies against the confidences (or simply against the centers of the bins), see Figure 1. If the classifier is confidence calibrated and the estimates of acc_m are good enough (which depends on the number of samples in the bin), the reliability curve will be almost diagonal.

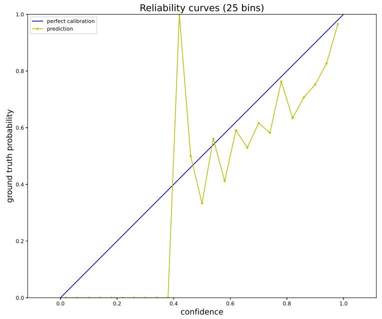

Figure 1. Prediction reliability curve of a small MLP trained on a synthetic data set. In the middle we see large deviations from the diagonal

Unfortunately, a lot of important information is hidden in this kind of visualization. In particular, the weight factors $N_m / |D|$ are not displayed. In other words, each bin contributes equally to the plot, independently of how many samples fall into it. This information can be added with a histogram of the confidences, as shown in Figure 1.

3.2 Other binning schemes

Estimators based on a fixed number of equally sized bins, despite being simple to implement and intuitively clear, have multiple downsides. One of them is the previously mentioned loss of information in the resulting reliability curve. Another one is that they may have a significant bias, depending on the distribution from which D was sampled. Examples showing this bias were constructed in [VWA+].

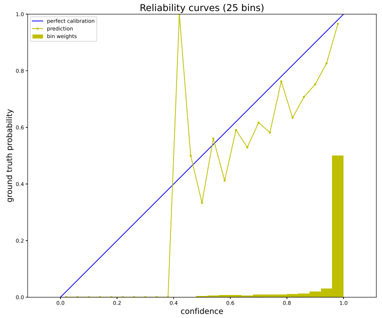

Figure 2. Same curve as in Figure 1 but with an added histogram of confidences. We see that most of the bins in the middle are almost empty (in comparison) and hence don't contribute much to the ECE

Generally, the estimate of the ECE will heavily depend on the binning scheme, especially when D is small. Several binning schemes that are adapted to the data have been proposed in the last years, for example in [NDZ+] and [DLXS]. If in such an adaptive scheme the bins have an equal number of samples, the resulting reliability curve is easier to interpret and there is no need to plot the bin populations as done above.

3.3 A note on variance and convergence

Since randomness is involved in the selection of test sets, an analysis of variance and convergence rate of the estimators is required to make statistically significant statements. Several methods for such an analysis are based on the idea of *consistency resampling* [BS], where one assumes that the classifier is strongly calibrated and samples new labels (not real ground truth labels) from categorical distributions constructed from the classifier's outputs. Repeating this procedure multiple times allows to compute confidence intervals for the per-bin accuracies and hence to assess whether the observed value of the estimator or the observed reliability curves are consistent with strong calibration.

While there is a lot more that must be said about the statistical properties of calibration metrics, it is beyond the scope of this introductory review.

4 Measuring calibration through simulation

Even if one has an application in which calibration does play a role, it might still not be clear how relevant it is for an accurate classifier. In other words, given a classifier that predicts correctly in, say, 90% of the cases, how important is it for the application that this classifier is also calibrated and which calibration metrics are the relevant ones?

One way to evaluate this is to identify the quantity one wishes to optimize and to analyze the effect of miscalibration of the classifier while maintaining accuracy. This quantity may be the expected gain in value in a financial application, where a mixed strategy based on predicted confidences is applied, or the expected percentage of successful treatments in a medical application, based on confidences in the diagnosis.[10] The confidence-based decision process can then be simulated and probabilistic classifiers can be compared to each other directly in an application-centric context.

There are two fundamentally different situations in which this procedure can be useful: *requirements engineering* and *model evaluation*. Let us view these situations through the eyes of a product owner (PO) who is to assign the task of developing a probabilistic classifier to a team of experts. The PO needs to specify in advance how well the classifier must perform to be acceptable (including calibration), and she also needs a method to evaluate the classifier on a test set. A simulated decision process helps in both situations.

Requirements engineering with fake classifiers. At this stage there is no model yet and the goal is to understand the minimal requirements on the model quality needed to reach the application's target performance. Having the simulated decision making process at hand, one could use a *fake classifier*, i.e. a generator of confidence vectors and ground truth labels that is not based on actual data (and does not even take input), but has *adjustable accuracy and calibration.* Using such fake classifiers one could simply compute how sensitive the quality of decision making is to accuracy and calibration without actually training any models. This is a cheap and efficient method of finding

[10] appliedAI has developed *KALE, the calibration game*, to illustrate this situation in an idealized problem of resource management in healthcare

out which properties of calibration matter in the specific context of interest. We say more about such fake classifiers in Section 6.

Model evaluation on test sets. Once classifiers have been trained, accuracy and calibration metrics can be computed on test sets. As mentioned before, it is often far from trivial to interpret the various calibration errors when one is concerned with performance in a specific context. Instead, one could evaluate models using the same simulated decision process as for requirements engineering, running it on the test set with the real classifier.

5 Recalibrating

The calibration of a classifier can be improved either by adjusting the training or modeling, or by recalibrating via post-processing using a held-out set. Examples of the former include training an ensemble of models [LPB] or using specialized loss functions [MKS+].

Here we focus on the second approach, i.e. recalibrating with a held-out set. The practical advantage of these methods is that they can be applied to any classifier and, by definition, do not require re-training. Since the classifier itself is not modified in this approach, we can forget about the actual inputs and directly consider the distribution of confidence vectors obtained on the test set. We are thus concerned with the problem of having samples of true labels and confidences, following a miscalibrated distribution Y, C with $C := c(X)$, and want to find a **calibration function** $r_\theta : \Delta^{K-1} \longrightarrow \Delta^{K-1}$ such that $Y | r_\theta(C)$ is better calibrated (see Section 1.3 for more details). Note that, as long as one ensures that for every $c \in \Delta^{K-1}$ it holds that

$$\operatorname{argmax}(r_\theta(c)) = \operatorname{argmax}(c),$$

then mapping the output confidences with r_θ does not have an effect on the model's accuracy.

There are a number of different methods, parametric and non-parametric, for constructing or learning calibration functions. We will only highlight two here: *temperature scaling* and an extension thereof called *Dirichlet calibration*. For a summary of other techniques, see [GPSW, KPK+].

5.1 Temperature scaling

Temperature scaling was first applied to multi-class problems in [GPSW]. For concreteness, consider a standard neural network classifier. The idea and implementation are fairly straightforward: scale the output logits of the network with a learned parameter prior to applying the softmax. The parameter is expressed as $1/T$ and T is called **tem-**

perature due to analogies in statistical physics. T can be learned by minimizing the negative log likelihood (NLL) over the held-out set. Note that scaling the logits uniformly does not alter the ranking of predictions, thus accuracy is unaffected by it. **Temperature scaling** proposes a calibration function of the form:

$$r_T(c)=\sigma\left(\frac{\log(c)}{T}\right),$$

where the logarithm acts component-wise and σ is the softmax function, although in implementations the scaling should rather act directly on the predictor's logits, before the application of the softmax that converted them to confidences.

Temperature scaling has been empirically shown to be efficient in reducing the ECE of neural networks. Unfortunately, it may underperform in improving overall calibration, e.g. in reducing class-wise calibration errors. This is where Dirichlet calibration is reported to perform better

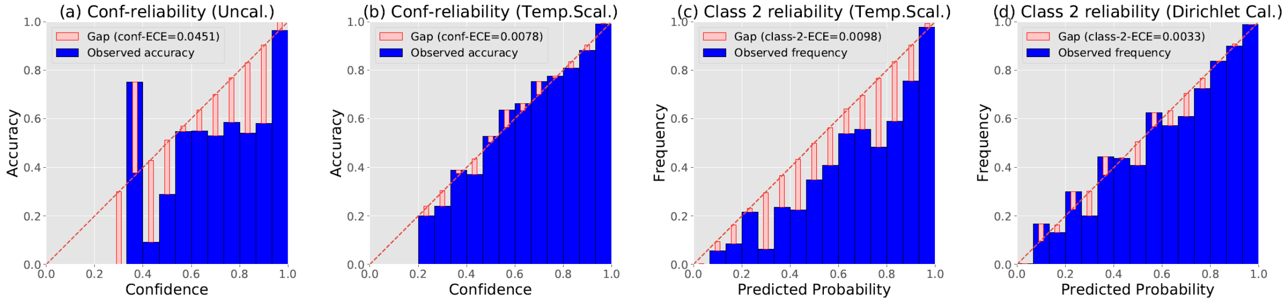

Figure 3. Reliability diagrams of `c10_resnet_wide32` on CIFAR-10: (a) confidence-reliability before calibration; (b) confidence-reliability after temperature scaling; (c) classwise-reliability for class 2 after temperature scaling; (d) classwise-reliability for class 2 after Dirichlet calibration. Image and description taken from [KPK+]

5.2 Dirichlet calibration

A straightforward extension of temperature scaling is to multiply the logits by a weight matrix $W\in\mathbb{R}^{K\times K}$ and to add a bias vector $b\in\mathbb{R}^{K}$. In other words, this extension assumes a calibration function of the form

$$r_{W,b}(c)=\sigma(W\cdot\log(c)+b). \tag{5}$$

Again, in implementations one should use logits instead of taking the log of C, in which case the procedure is also known as *matrix scaling* (theoretically, the two are equivalent but in practice matrix scaling performs slightly better, presumably due to rounding errors when calculating log of softmax of logits). Equation (5) can be shown to follow from the assumption that confidences conditioned on ground truth are distributed with Dirichlet distributions, i.e. that there are K vectors with positive entries $\alpha^{(k)} = \left(\alpha_1^{(k)}, \ldots, \alpha_K^{(k)}\right)$ such that for each $k \in \{1, \ldots, K\}$:

$$C \mid Y = k \quad \sim \quad \mathrm{Dir}(\alpha^{(k)}),$$

hence the name, **Dirichlet calibration**. This recalibration scheme has two downsides: it can easily overfit due to the number of parameters (quadratic in number of classes) and it does not preserve ranking and accuracy of predictions. The latter might actually not be a problem as recalibration also has the potential to increase accuracy, especially if trained to optimize NLL. However, one needs to be more careful if using a different objective.

In the initial experiments with matrix scaling in [GPSW] the authors encountered severe overfitting, even with regularization. However, with a recently proposed regularization scheme that separately regularizes off-diagonal elements of W and the intercept b (giving rise to the name ODIR - off-diagonal and intercept regularization), matrix scaling outperformed temperature scaling in most cases with up to 100 classes (see [KPK+] for details). Importantly, it improved calibration not only as measured by ECE but also by class-wise calibration errors - where temperature scaling performed significantly worse.

6 Fake classifiers

We have mentioned that for requirements engineering it may be useful to create fake confidence vectors and ground truth labels that have a pre-determined accuracy and calibration. By fake we mean that instead of being computed by applying a classifier to data, the confidences and labels are simply sampled from a distribution.

Such fake classifiers are also useful for the evaluation of tools in calibration, like metrics and recalibration algorithms. For metrics, one can empirically study convergence rates since, typically, for a fake classifier the true value of a calibration metric is a known integral. Recalibration algorithms can be benchmarked with fake classifiers. This is important since miscalibration can arise in a number of different ways: classifiers can be overconfident or under-confident in predictions, be overconfident for some classes and under-confident for others or have a mixture of different behaviors. To the best of our knowledge, recalibration algorithms have been studied up to now by applying them to models like deep neural networks trained on selected data sets, like CIFAR, MNIST and IMAGENET. Fake classifiers enable the analysis of recalibration for a much more diverse variety of scenarios, and in a controlled way.

However, creating fake classifiers where accuracy and calibration can be adjusted mostly independently of each other is not entirely trivial. This construction is explained in detail in [PBfD] where the results of benchmarking several algorithms and metrics will be presented as well. Accompanying the paper, we will open-source the code for creating fake classifiers and performing benchmarking as part of the aforementioned library [PSB] containing general tools for calibration.

7 Accuracy and calibration

We already mentioned that accuracy and calibration are up to a point independent from each other. However, for an important special case, there is a relation between accuracy and ECE. This case is that of *always overconfident or under-confident classifiers.*[11] Since modern neural networks tend to be always overconfident and ECE is one of the most widely used calibration metrics, this relation is of practical importance. In [PBfD], we prove:

Lemma 1.

i. *For always overconfident classifiers, it holds that:*

$$\text{ECE} \leq 1 - \text{acc}.$$

ii. *For always underconfident classifiers, it holds that:*

$$\text{ECE} \leq \text{acc}.$$

While the second inequality is not very interesting (for an accurate classifier it is a very loose bound), the first one is tighter and makes intuitive sense: an overconfident classifier that is very accurate simply does not have much room for miscalibration. In fact, a perfectly accurate classifier cannot be overconfident at all - at perfect calibration it will always predict confidence vectors with 1 at the predicted class and 0 everywhere else. This inequality holds for many neural networks and shows that it might be enough to only improve accuracy - the ECE will be automatically small when the accuracy is very high. For requirements engineering in situations where the ECE is a relevant metric, this bound should be taken into consideration.

[11] "Always" means that the confidence $c^* = \max(c(x))$ of every prediction $y = \operatorname{argmax} c(x)$ is greater than, resp. lower than, the value $\mathbb{P}(Y = y | c(X) = c^*)$

BIBLIOGRAPHY

[BS] Jochen Bröcker and Leonard A. Smith. Increasing the Reliability of Reliability Diagrams. 22(3):651–661.

[DLXS] Yukun Ding, Jinglan Liu, Jinjun Xiong, and Yiyu Shi. Revisiting the Evaluation of Uncertainty Estimation and Its Application to Explore Model Complexity-Uncertainty Trade-Off.

[GPSW] 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*, pages 1321–1330.

[Kle] Achim Klenke. *Probability Theory. A Comprehensive Course*. Universitext. Springer London, 2 edition.

[KPK+] 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 H. Wallach, H. Larochelle, A. Beygelzimer, prefix=d\textquotesingle, useprefix=false family=Alché-Buc, given=F., E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 12316–12326. Curran Associates, Inc.

[LPB] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6402–6413. Curran Associates, Inc.

[MKS+] Jishnu Mukhoti, Viveka Kulharia, Amartya Sanyal, Stuart Golodetz, Philip Torr, and Puneet Dokania. Calibrating deep neural networks using focal loss. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 15288–15299. Curran Associates, Inc.

[NDZ+] Jeremy Nixon, Michael W Dusenberry, Linchuan Zhang, Ghassen Jerfel, and Dustin Tran. Measuring Calibration in Deep Learning. Page 14.

[PBfD] Michael Panchenko, Anes Benmerzoug, and prefix=de, useprefix=true family=Benito Delgado, given=Miguel. Class-wise and reduced calibration methods. IEEE.

[PSB] Michael Panchenko, Lorenzo Sticher, and Anes Benmerzoug. Kyle: A toolkit for classifier calibration.

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

[WLZ] David Widmann, Fredrik Lindsten, and Dave Zachariah. Calibration tests in multi-class classification: A unifying framework. In H. Wallach, H. Larochelle, A. Beygelzimer, prefix=d\textquotesingle, useprefix=false family=Alché-Buc, given=F., E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 12257–12267. Curran Associates, Inc.