Sample-Dependent Adaptive Temperature Scaling for Improved Calibration

Tom Joy[1,3], **Francesco Pinto**[1], **Ser-Nam Lim**[2], **Philip H. S. Torr**[1], **Puneet K. Dokania**[1,3]
[1]University of Oxford,
[2] Meta AI,
[3]Five AI
tom.joy@five.ai

Abstract

It is now well known that neural networks can be wrong with high confidence in their predictions, leading to poor calibration. The most common post-hoc approach to compensate for this is to perform temperature scaling, which adjusts the confidences of the predictions on any input by scaling the logits by a fixed value. Whilst this approach typically improves the average calibration across the *whole* test dataset, this improvement typically reduces the individual confidences of the predictions irrespective of whether the classification of a given input is correct or incorrect. With this insight, we base our method on the observation that different samples contribute to the calibration error by varying amounts, with some needing to increase their confidence and others needing to decrease it. Therefore, for each input, we propose to predict a different temperature value, allowing us to adjust the mismatch between confidence and accuracy at a finer granularity. Our method is applied post-hoc, enabling it to be very fast with a negligible memory footprint and is applied to off-the-shelf pre-trained classifiers. We test our method on the ResNet50 and WideResNet28-10 architectures using the CIFAR10/100 and Tiny-ImageNet datasets, showing that producing per-data-point temperatures improves the expected calibration error across the whole test set.

1 Introduction

For neural networks (NNs) to be employed in real-world safety-critical applications, we do not only require them to produce correct predictions, but also provide reliable confidence estimates in their predictions (i.e. they are calibrated). Limiting our scope to neural classifiers, using the maximum probability of the predictive distribution as a confidence measure, literature has established that a mismatch exists between such notion of confidence and the expected accuracy. Indeed, such models generally suffer from being *on average* overconfident over the test set.

A simple approach to rectify this issue is to perform *temperature scaling* (Guo et al. 2017), a post-hoc method which scales the logits by a single scalar value, obtained through cross validation. This approach improves the classifier's performance on standard calibration metrics across a test set. However, from a per-sample point of view there are significant issues. Since the temperature is found by minimising the calibration error (in expectation) over the *entire* validation set, and since neural networks are overconfident on average, practically speaking, the effect of temperature scaling is to reduce the confidence for every prediction. However, as we will discuss, different samples contribute by varying amounts to the calibration error.

This issue can be seen in Fig. 1, which shows the histogram of the individual contributions to the calibration errors; i.e. the distribution of $p(\mathbf{y}|\boldsymbol{p}_i) - \boldsymbol{p}_i$, where $p(\mathbf{y}|\boldsymbol{p}_i)$ is the accuracy and $\boldsymbol{p}_i$ is the softmax probability for the data point i, the calibration error can be obtained by taking the weighted average over all the values.[1] Here the mismatch between per data-point confidence and accuracy is not constant across all the data-points, and hence miscalibration cannot be fixed by scaling the logits by a single fixed value, a key assumption in vanilla temperature scaling. The calibration error varies significantly, with a small (but not insignificant) number of samples on which the network is overconfident. Consequently, scaling the predictions with a single temperature value will adjust *all* of the errors in the same way. Typically, the temperature values obtained are greater than 1, resulting in a reduction of confidence of *all* predictions, regardless of whether they are correct with low confidence or incorrect with high confidence.

To combat this, we propose a method which produces per-data-point predictions of the temperature, permitting an adequate decrease in the confidence on samples which the classifier is *likely* to get wrong, and an increase in the confidence on predictions it is *likely* to get correct. As a result, we obtain better test Expected Calibration Error (ECE)(Guo et al. 2017) both on in-distribution sets (i.e. the test set is i.i.d. with respect to the training set) and under covariate-shifted sets (i.e. the test set shares the same set of labels of the training set, but the inputs are not i.i.d. with respect to the training set).

Like temperature scaling, our method is applied post-hoc and is very fast to train and test. We extensively test the calibration of ResNet50 (He et al. 2016) and WideResNet28 (Zagoruyko and Komodakis 2016) when using our method on CIFAR10/CIFAR100 and TinyImageNet, in-

[1]Here $p(\mathbf{y}|\boldsymbol{p}_i)$ is obtained through histogram binning and represents the accuracy of each bin, and the weights are proportional to the number of samples in each bin.

cluding results under data-shift (Hendrycks and Dietterich 2019).

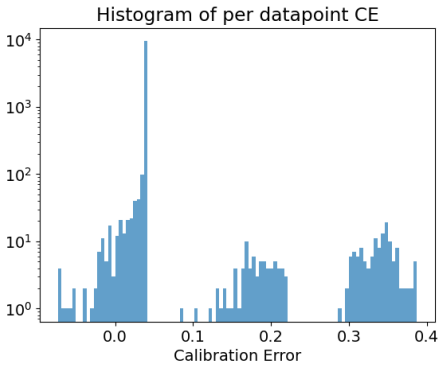

Figure 1: Histogram of per sample contribution to calibration error, positive numbers indicate overconfidence. Here we can see that the samples contribute by different amounts to the overall calibration error. Predictions are for CIFAR-10.

2 Problem Formulation

Network Overconfidence and Temperature Scaling

Given an input $\mathbf{x}$, a standard K-class neural classifier first extracts a feature embedding $\Phi(\mathbf{x})$ before computing the logits $\mathbf{s} = f(\Phi(\mathbf{x})) \in \mathbb{R}^K$ and finally applying the softmax operator $\boldsymbol{p} = \sigma(\mathbf{s}) = \frac{\exp(\mathbf{s})}{\sum_i \exp(s_i)}$ to obtain the class probabilities for the categorical distribution, the prediction is then given as $\hat{y} = \arg\max_k p_k$, where $\boldsymbol{p} = \{p_k\}^K$. A classifier is said to be calibrated if the confidence in its prediction (usually taken to be $\max_k p_k$) matches its accuracy on expectation, i.e. if a classifier makes predictions with a confidence of 80% for a certain set of points, then the accuracy should be 80% on the set of points. Typically, neural networks are overconfident as the confidence of the predictions are higher than their expected accuracy (Guo et al. 2017).

Temperature scaling (Guo et al. 2017) consists of re-scaling the logits by a constant factor $T \in \mathbb{R}^+$ before applying the softmax, i.e. $\boldsymbol{p}' = \sigma(\frac{\mathbf{s}}{T})$. The value of T can drastically affect the entropy of the predicted distribution, which is demonstrated in Fig. 2, where a value of $T > 1$ leads to a higher entropy distribution (the higher T, the higher the entropy); a value of $T < 1$ leads to a lower entropy distribution (the lower T, the more "peaky" the distribution).

The temperature T is usually found by minimising the ECE or the Negative Log-Likelihood (NLL) using a validation set. Typical optimal values for T are usually greater than 1 (Mukhoti et al. 2020), indicating that, on average, optimising the ECE or NLL across the validation set leads to a higher entropy of the predictions. However, this approach decreases the confidences of *all* the predictions without considering that the miscalibration error can vary widely on a data-point basis. For correct predictions, temperature scaling will make the predictions more under-confident, whilst for incorrect predictions, the temperature may not be the right value to bring the confidences down to a level which will make it calibrated.

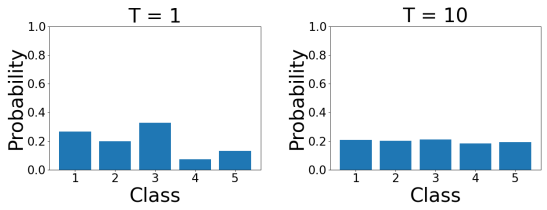

Figure 2: Example plots of Softmax distribution with different temperature values for fixed logits. Left to right: $T = 1.0$ and $T = 10.0$.

Loosely speaking, this suggests that further improvements in calibration can be achieved by using a variable temperature T, predicted on a per data-point basis (i.e. $T = g(\mathbf{x})$), permitting $T > 1$ for samples which are correctly classified, and $T < 1$ for the incorrect ones. This enables the model to have a greater flexibility when compensating for miscalibration, as it respects the individual contributions each sample makes to the ECE. Moreover, this approach can be applied *without* affecting a classifiers accuracy.

3 Issues with Jointly Learning Temperature Alongside the Network Weights

One might suggest introducing the temperature as one of the outputs of the network and jointly learning it as part of the training process. Here we outline why this approach of learning to predict the temperature values T and the predictive probabilities $\boldsymbol{p}$ might result in suboptimal performance. Consider the last layer of a NN with parameters $\boldsymbol{w} \in \mathbb{R}^{D \times K}$ for a feature space of size D and the cross entropy loss $\mathcal{L} : \mathbb{R}^K \rightarrow \mathbb{R}$. The gradient for the layer is given as

$$\frac{\partial \mathcal{L}}{\partial \bar{\boldsymbol{w}}} = \frac{\partial \mathbf{s}}{\partial \bar{\boldsymbol{w}}}(\sigma(\mathbf{s}) - \mathbf{y}), \tag{1}$$

where $\mathbf{y}$ denotes the one-hot ground-truth label and $\sigma(\mathbf{s}) - \mathbf{y} = \{\sigma(s_j) - y_j : j \in \{1 \dots K\}\}$, $\bar{\boldsymbol{w}}$ indicates the network weights are flattened to column vector form. Inspecting the gradients indicates that the gradient starts to vanish when $s_k \rightarrow \infty$ and $s_{\setminus k} \rightarrow -\infty$, where k is the correct class. Or to put it simply, the optimisation does not converge until the network produces one-hot logits.

This forces the magnification of the network weights (Mukhoti et al. 2020), which subsequently leads to an over-confident network and hence miscalibrated predictions. A mechanism to achieve the desired one-hot prediction without magnifying the weights could be instead to naïvely learn the temperature alongside the logits, assuming the model is trainable and converges. In this case, gradient updates would decrease the value of T, resulting in a lower-entropy distribution that is more "peaky". We now discuss why this approach might not work well in practice.

If we consider the gradient of the temperature, which is

given as

$$\frac{\partial \mathcal{L}}{\partial T} = \sum_k \frac{y_k}{T^2}\left(s_k \sum_{i \setminus k} \exp\left(\frac{s_i}{T}\right) - \sum_{j \setminus k} s_j \exp\left(\frac{s_j}{T}\right)\right), \tag{2}$$

which decreases the value of T for a correct prediction ($\tilde{k} = \arg\max_k s_k$), leading to more confident predictions. Typically, the train accuracy will approach 100%, meaning that gradient updates to T cause it to decrease without any moderation, preventing the network from learning how to predict T appropriately. In short, there is essentially only data for correct predictions, *preventing crucial information on how the network should behave when it's wrong being incorporated in the learning process*. Consequently, learning T naïvely is not a feasible option as the network just learns to be confident everywhere.

Learning to Calibrate

Our approach involves a new temperature prediction module (a small neural network) that operates on each input sample independently and whose objective is to extract information from the trained model itself in order to calibrate the confidences of each prediction. We call this learning to calibrate.

Doing so requires learning a temperature prediction module on a data-set consisting of data-points $\mathcal{X}_{cal} = \{\mathbf{x}_n\}^N, \mathcal{X}_{train} \cap \mathcal{X}_{cal} = \emptyset$, neural network predictions $\mathcal{P}_{cal} = \{\boldsymbol{p}_n\}^N$ and labels $\mathcal{Y}_{cal} = \{y_n\}^N, y \in \{, \dots, K\}$ for K classes. It is important to note that the objective here is to learn to assign low confidences to data points which are *likely* to be incorrect and high confidences to those which are *likely* to be correct.

Specifically for a given data-point $\mathbf{x} \in \mathcal{X}_{cal}$, we propose to optimise the temperature prediction module over T by maximising the log probability of the label $\mathbf{y}$ under the Categorical probability distribution parametrised by the T-scaled logits $\mathbf{s}$, i.e. $T^* = \arg\max_T \log \text{Cat}(y; \text{softmax}(\mathbf{s}/T))$[2]. Here we do not optmise $\mathbf{s}$ but keep it fixed; we are only optimising w.r.t to T.

In situations where $y = \arg\max_k \boldsymbol{p}_k$ (i.e. correct prediction), the target function is maximised when $T \to 0$, as we want the predicted probabilities to match the one-hot logits, e.g. see $T = 0.1$ in Fig. 2. This is equivalent to minimising the entropy of the predictive distribution by only manipulating T, which is the desired outcome for a correct prediction.

In situations where the prediction is incorrect, $y \neq \arg\max_k \boldsymbol{p}_k$, to maximise the target function we need to maximise $\boldsymbol{p}_{\mathbf{y}}$ and minimise $\boldsymbol{p}_{\tilde{k}}$, where $\tilde{k} = \arg\max_k \boldsymbol{p}_k$. As the temperature prediction module cannot change the predicted label, the optimzation accepts the incorrect prediction and maximise the target function by flattening the Softmax outputs with $T >> 1$, which is equivalent to maximising the entropy of the predictive distribution. This effect can be seen by considering the case where predicting class 2 in Fig. 2 is the incorrect prediction; among the three cases shown, $T = 10$ maximises $\text{Cat}(y \neq 2; \text{softmax}(\mathbf{s}/T))$.

[2]Which is equivalent to the cross entropy loss.

4 Adaptive Temperature Scaling

We are now ready to outline the specifics of our proposed temperature prediction module. Given a data-set $\mathcal{X}_{cal}$, we want to learn which samples the classifier should be confident about and which it should not. Rather than acting on the image space, we instead use the feature extractor of the classifier, as it has already learnt how to extract the information needed for class prediction and also contains a notion of the associated confidence. This has been demonstrated in (Guo et al. 2017), where working only on the feature space has already provided highly promising results in calibrating models for a variety of tasks; suggesting that the feature space already contains sufficient information for calibration. What we now need is a method to extract this confidence information from the feature space and leverage it appropriately to calibrate the predictions.

Representing Uncertainty with the Variational Autoencoder variational autoencoders (VAEs) (Kingma and Welling 2013) act as an efficient model to obtain representations of data; the representations encapsulate the generative factors in a lower-dimensional subspace and are rich enough to reconstruct the data sample. In the specification of the generative model, the user has to specify a prior over the latent variables (typically an isotropic Gaussian) where the KL distance between the prior and approximate posterior is minimised during training. Unlike a standard autoencoder (Hinton and Zemel 1994), there is now a mechanism to obtain a likelihood on the latent codes. In reality this value forms part of the importance weight and can be used as a proxy to the true likelihood but avoids issues associated with deep generative models (Nalisnick et al. 2019).

From a mechanistic point of view, we expect samples which are much more common to be placed in the centre of the prior. Here we leverage this idea and use the latent likelihoods as a basis to predict the temperature value. Indeed, we find empirically that this approach works well in practice. Rather than using an isotropic Guassian as the prior, we instead introduce a Gaussian mixture prior, with component for each class specified by the *learnable* parameters $\lambda_k = \{\mu_k, \sigma_k\} \in \mathbb{R}^{D_{\mathbf{z}}}$, such that $p_{\boldsymbol{\lambda}}(\mathbf{z}) = \frac{1}{K}\sum_y p_{\lambda_y}(\mathbf{z} \mid y)$. This allows for an individual unimodal prior for each class; preventing any issue with clusters for individual classes being placed in lower likelihood regions of the latent space, as would be the case if an isotropic prior is used. With this mixture prior, the evidence lower bound is given as

$$\log p(\Phi(\mathbf{x}), y) \geq \mathbb{ELBO}[\Phi(\mathbf{x}), y] = \mathbb{E}_{q_\varphi(\mathbf{z}|\Phi(\mathbf{x}))} \log \frac{p_\vartheta(\Phi(\mathbf{x})|\mathbf{z}) p_{\lambda_y}(\mathbf{z} \mid y)}{q_\varphi(\mathbf{z}|\Phi(\mathbf{x}))}, \tag{3}$$

where $q_\varphi(\mathbf{z}|\Phi(\mathbf{x}))$, $p_\vartheta(\Phi(\mathbf{x})|\mathbf{z})$ and $p_{\lambda_y}(\mathbf{z} \mid y)$ represent the encoder, decoder and mixture prior component, the parameters of the VAE are given as $\boldsymbol{\Theta} = \{\vartheta, \varphi, \lambda_1, \dots, \lambda_K\}$. The parameters of the mixture prior are learnt alongside the parameters of the encoder and decoder(Tomczak and Welling 2018). The use of this mixture prior forces the aggregate posterior for *each* class to match a Gaussian distribution, i.e. $\sum_{\mathbf{x} \in \mathcal{X}_k} q(\mathbf{z}|\Phi(\mathbf{x})) \approx \mathcal{N}(\mathbf{z}; \mu_k, \sigma_k)$. This encourages the

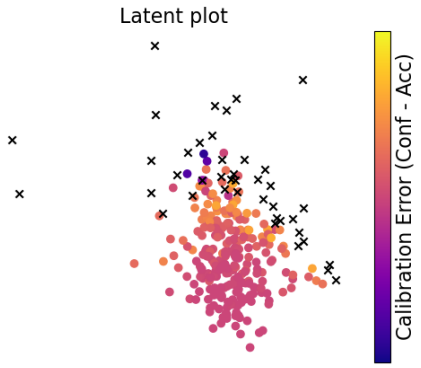

Figure 3: t-SNE plot for classes `cat`, colour indicates per data-point contribution to ECE, pink indicates no contribution to ECE, yellow is positive contribution, purple is negative. Generally, samples with little contribution to calibration error (pink) are placed around the centre of the cluster, unlike samples with a high contribution (yellow and purple) which are placed near the edges. Furthermore, incorrect samples (black cross) are placed significantly far away from the cluster centre.

representations for each class to cluster around a known distribution $p_{\lambda_y}(\mathbf{z} \mid y)$, which we will use to obtain a pseudo likelihood to predict the temperature value. The choice of the VAE was in part down to the motivation that samples which contribute significantly to the ECE will have lower latent-likelihood but also because empirically we found it worked well in practice. Before outlining the details of the approach in the next two subsections, we first provide evidence of this empirical motivation to use a VAE.

We now perform a preliminary experiment, which serves to investigate which samples in the latent representation contribute the most to the calibration error. Specifically, we construct a t-SNE plot for each class of CIFAR-10 but colour code the points depending on their per-data-point contributions to the calibration error. This provides a visual method for us to inspect where samples which harm calibration are placed, which can be seen in Fig. 3. Here we can see that data-points which do not contribute to the calibration error tend to be placed near the centre of the cluster, and ones which do, or are incorrect, indicated by a black cross, are placed far from the centre. This highlights that the VAE is able, to some extent, to provide a basis to predict the temperature, we then utilise this representation to predict T through a simple Multi Layer Perceptron.

Temperature Prediction Network

Given that the VAE structures the latent in space in a way which makes it amendable to confidence prediction, we learn a very simple MLP parameterised by θ, which predicts the temperature based on the latent embeddings, using the cross entropy loss as an objective. Rather than using the latent samples as input to the MLP, given the observations in Fig. 3, we choose to predict the temperature as a function of the vector of log-likelihoods on *all* of the conditional priors, specifically $T = g_\theta(\tilde{\boldsymbol{q}})$ where $g : \mathbb{R}^K \rightarrow \mathbb{R}$ is the MLP which predicts the temperature and $\tilde{\boldsymbol{q}} = \{\log p_{\lambda_y}(\mathbf{z} \mid y) | \forall y\}$, i.e each element $\tilde{\boldsymbol{q}}_i$ contains the log-likelihood of $\mathbf{z}$ on the corresponding conditional prior $p_{\lambda_y}(\mathbf{z} \mid y)$. Evaluating $\log p_{\lambda_y}(\mathbf{z} \mid y)$ can be viewed as a pseudo likelihood of $\mathbf{x}$, consequently the module predicts the temperature as a non-linear transform of a pseudo-likelihood of the sample. It is also important to point out that due to the use of feature space as the input, we are able to use small architectures, making this approach very fast during training and at test time. We represent a high level overview and the graphical model in Fig. 4.

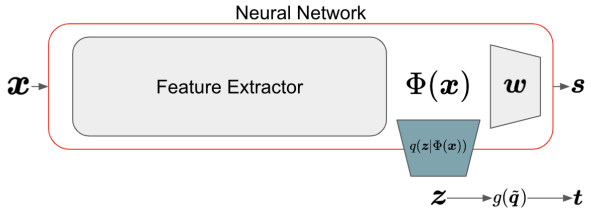

Figure 4: High level architecture. The off shelf neural-network is represented by the red box, where the parameters are left unchanged, the learnable VAE encoder is indicated by $q(\mathbf{z}|\Phi(\mathbf{x}))$, with the $g_\theta(\tilde{\boldsymbol{q}})$ as the MLP predicting T.

Calibrated Training Details

The overall post-hoc learning algorithm is very simple and the module can be trained in *under a minute* on an 8Gb Titan Xp for most datasets, depending on the validation set and feature space size, we give an overview of the procedure in Alg. 1. We combine learning the VAE and the temperature prediction network into one objective. Specifically, we maximise the following objective

$$\mathcal{L}(\mathbf{x}, y) = \mathbb{ELBO}[\Phi(\mathbf{x})] + \log \text{Cat}(y \mid \text{softmax}(\mathbf{s}/g_\theta(\tilde{\boldsymbol{q}})) \tag{4}$$

with $\tilde{\boldsymbol{q}} = \{\log p_{\lambda_y}(\mathbf{z} \mid y) | \forall y\}$ $\mathbf{z} \sim q_\phi(\mathbf{z} \mid \mathbf{x})$ and using Normal distributions for $\mathbf{z}$; Laplace distribution over $\Phi(\mathbf{x})$ (L1 loss); and a Categorical over $\mathbf{y}$. To train the VAE, we use a held-out dataset from training the network, i.e. $\mathcal{X}_{train} \cap \mathcal{X}_{cal} = \emptyset$. We used the Adam optimiser with a learning rate of 0.001 and trained for 50 epochs. This is an additional benefit of training the VAE on the $\Phi(\mathbf{x})$, as the feature space has a lower dimensionallity and simpler structure than the image space, leading to much faster training and the ability to use simpler networks.

Calibration at Test Time During test time, the features of the data point $\Phi(\mathbf{x})$ and predicted logits are computed using the classifier $\mathbf{s} = f(\Phi(\mathbf{x}))$, the temperature can then be predicted through $T = g_\theta(\tilde{\boldsymbol{q}})$, where $\tilde{\boldsymbol{q}} = \{\log p_{\lambda_y}(\mathbf{z} \mid y) | \forall y\}$ with $\mathbf{z} = q_\phi(\mathbf{z} \mid \mathbf{x})$. The calibrated predictions are then computed as $\boldsymbol{p} = \sigma(\mathbf{s}/T)$.

Algorithm 1: Learning Adaptive Temperature

Require: $\mathcal{X}_{cal}, \mathcal{Y}_{cal}, \mathcal{P}_{cal}$

while not converged **do**
  $\mathbf{x}, \mathbf{y} \leftarrow$ Random batch
  $\nabla_{VAE} \leftarrow \nabla \mathbb{ELBO}[\Phi(\mathbf{x})]$
  $\tilde{\boldsymbol{q}} = \{\log p_{\lambda_y}(\mathbf{z} \mid y) | \forall y\}\}$ $\mathbf{z} \sim q(\mathbf{z}|\Phi(\mathbf{x}))$
  $\nabla_T \leftarrow \nabla \log \text{Cat}(\mathbf{y}; \text{softmax}(\mathbf{s}/g_\theta(\tilde{\boldsymbol{q}}))$
  $\{\boldsymbol{\Theta}, \theta\}_{t+1} \leftarrow \{\boldsymbol{\Theta}, \theta\}_t - \alpha(\nabla_{VAE} + \nabla_T)$
end while

5 Results

Before evaluating the model, we define the hypothesis we are trying to test. Specifically, we want to evaluate if predicting the temperature on a per-data-point basis leads to improved calibration over vanilla temperature scaling. Secondly, we wish to investigate how adaptive temperature performs under dataset shift.

We performed our experiments on the WideResNet28-10 (Zagoruyko and Komodakis 2016) architecture[3]. We report calibration results on CIFAR10/CIFAR100 (Krizhevsky, Hinton et al. 2009) and Tiny-ImageNet (Torralba, Fergus, and Freeman 2008). We conducted distribution-shit experiments using the variants CIFAR10-C/CIFAR100-C to test for domain shift (Hendrycks and Dietterich 2019). We used the following as models for our evaluation:

- Cross Entropy Loss, due to it's popularity and wide adoption.
- Brier Score (Brier et al. 1950), due to it's ability to obtain well calibrated predictions (Mukhoti et al. 2020).
- Deep Ensembles (Lakshminarayanan, Pritzel, and Blundell 2016), as it achieves state of the art results.[4]

Results are obtained for multiple seeds for Cross Entropy and Brier Score, but only one seed for Deep Ensembles due to the number of models needed.

Calibration

Here we evaluate how adaptive temperature scaling affects standard calibration metrics compared to vanilla temperature scaling. We report results using the ECE, which divides the probability into equally sized bins and then computes the absolute difference between confidence and accuracy for each bin before taking the average. However, the ECE is known to be a biased estimator (Ding et al. 2020), with its performance depending on the binning size and on the distribution of samples in each bin. For this reason, the reliability of the ECE as a miscalibration metric is being questioned and several alternatives have been proposed (e.g. (Nixon et al. 2019; Roelofs et al. 2022; Mukhoti et al. 2020)). Among these, we choose to also use the AdaECE (Mukhoti et al. 2020), which uses adaptive bin sizes to ensure each bin contains the same number of samples.

[3]Results on ResNet50 (He et al. 2016) are in the Appendix.

[4]Applying adaptive temperature scaling to deep ensembles does not necessarily preseve accuracy, however as we show in our experiments the difference is negligible.

We report the results in Tab. 1 where it can be seen that adaptive temperature scaling improves calibration compared to standard temperate scaling. In all cases our method is able to outperform vanilla temperature scaling, with large improvements obtained when using the cross entropy loss, e.g. $0.93 \rightarrow 0.76$ and $3.76 \rightarrow 2.95$ ECE for CIFAR10 and CIFAR100 when using the WideResNet2810 Network[5].

Model	Scaling	Accuracy	ECE	AdaECE
		CIFAR10		
CE	-	95.52 ± 0.4	2.15 ± 0.1	2.13 ± 0.1
CE	TS	95.52 ± 0.4	0.93 ± 0.2	0.98 ± 0.3
CE	AdaTS	95.52 ± 0.4	$\mathbf{0.76 \pm 0.1}$	$\mathbf{0.86 \pm 0.2}$
Brier	-	95.84 ± 0.1	0.92 ± 0.1	1.50 ± 0.1
Brier	TS	95.84 ± 0.1	1.88 ± 0.2	1.94 ± 0.1
Brier	AdaTS	95.84 ± 0.1	$\mathbf{1.65 \pm 0.1}$	$\mathbf{1.61 \pm 0.1}$
Ensm	-	96.35	1.68	1.61
Ensm	VTS	96.35	0.61	0.68
Ensm	AdaTS	96.37	0.51	**0.46**
		CIFAR100		
CE	-	80.71 ± 0.1	5.76 ± 0.1	5.70 ± 0.1
CE	TS	80.71 ± 0.1	3.76 ± 0.2	3.68 ± 0.3
CE	AdaTS	80.71 ± 0.1	$\mathbf{2.95 \pm 0.4}$	$\mathbf{2.90 \pm 0.4}$
Brier	-	79.25 ± 0.1	4.19 ± 0.2	4.13 ± 0.2
Brier	TS	79.25 ± 0.1	$\mathbf{3.87 \pm 0.6}$	$\mathbf{3.90 \pm 0.6}$
Brier	AdaTS	79.25 ± 0.1	$\mathbf{3.67 \pm 0.8}$	$\mathbf{3.64 \pm 0.7}$
Ensm	-	83.19	4.24	4.21
Ensm	TS	83.18	3.71	3.55
Ensm	AdaTS	83.22	**2.95**	**2.66**
		Tiny-ImageNet		
CE	-	60.47 ± 0.1	7.54 ± 4.0	7.53 ± 4.0
CE	TS	60.47 ± 1.1	6.28 ± 2.4	6.15 ± 2.4
CE	AdaTS	60.04 ± 1.2	$\mathbf{5.18 \pm 1.4}$	$\mathbf{5.17 \pm 1.3}$
Brier	-	50.23 ± 0.4	5.56 ± 0.6	5.52 ± 0.6
Brier	TS	50.23 ± 0.4	$\mathbf{4.55 \pm 0.2}$	$\mathbf{4.43 \pm 0.6}$
Brier	AdaTS	50.23 ± 0.4	$\mathbf{4.43 \pm 0.4}$	$\mathbf{4.21 \pm 0.5}$
Ensm	-	66.16	6.21	6.19
Ensm	TS	66.16	5.12	5.06
Ensm	AdaTS	66.00	**4.58**	**4.41**

Table 1: Calibration results, here we can see that adaptive temperate scaling is able to improve calibration on a variety of models. Bold indicates the best result, or within one standard deviation of the best result.

Data-Shift A key hypothesis we want to test is how adaptive temperature scaling behaves under data-shift. Specifically, we use the widely used CIFAR10-C and CIFAR100-C datasets, which are corrupted versions of the CIFAR10 and CIFAR100 (Hendrycks and Dietterich 2019).

The dataset consists of standard CIFAR images which have undergone 15 synthetic corruptions (e.g. noise, weather conditions, image properties) at varying levels. Within this

[5]We include experiments for a standard autoencoder and MLP in the appendix

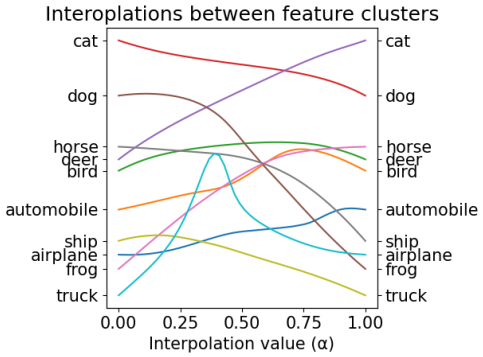

Figure 5: Left: How temperature varies when interpolating between class feature means on CIFAR-10. Right: Histogram of temperature values for each image in CIFAR-10, here we can see that typically objects have a lower temperature than animals, indicating they are easier to classify.

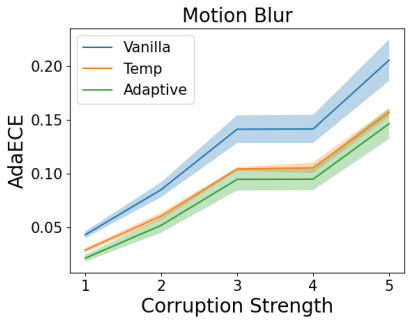

Figure 6: How AdaECE changes with varying levels of motion-blur corruptions on CIFAR-10. Adaptive temperature consistently produces lower error rates.

scenario, the classifier should either be robust to such corruptions (retaining accuracy) or if the accuracy is compromised, reduce the confidences accordingly. As such, we report the test accuracy as well as ECE and AdaECE in Tab. 2, where adaptive temperature scaling shows improvements over temperature scaling.

We also expect to see adaptive temperature scaling provide improvement over temperature scaling as the intensity of corruptions are increased for CIFAR-10-C. We generate plots highlighting the AdaECE calibration metric as the level of the corruption intensity is increased; the plot for `motion-blur` is displayed in Fig. 6. Here despite a general increase in error for all methods adaptive temperature scaling consistently produces lower error rates than vanilla temperature scaling (orange) and vanilla predictions (blue). More examples are provided in the Appendix.

Behaviour of the Temperature Prediction Module

Can the temperature module predict high temperature in uncertain regions? If yes, then we should see a change in the temperature as we traverse the feature the space[6]. To conduct this experiment, inspired by the analysis provided in (Pinto et al. 2021, 2022), we obtain the average feature representation for each class $\boldsymbol{\phi}_k = \frac{1}{|\mathcal{X}_k|}\sum_{\mathbf{x}\in\mathcal{X}_k}\Phi(\mathbf{x})$ and measure the temperature when interpolating between two classes. i.e. we predict the temperature for the features $\{\alpha\boldsymbol{\phi}_{k^{(i)}} + (1-\alpha)\boldsymbol{\phi}_{k^{(j)}}\} \quad \alpha \in [0,1]$. We plot the interpolation results in Fig. 5 (Left) for the classes in CIFAR-10, where the horizontal axis represent α and the vertical axis represents the temperature. For some classes we see a significant rise in the temperature as we interpolate between two classes, e.g. `automobile` and `bird`. This highlights the temperature prediction models ability to assign a low temperature in regions that the classifier is certain about, e.g. around the mean and a higher temperature in less certain regions, e.g. heavily interpolated regions.

Interestingly, this feature is not present for all class pairs; for some, e.g. `cat` and `dog`, where the temperature remains high between classes. We hypothesise that this is due to an interpolation between these classes being a plausible realisation of an image, unlike for `automobile` and `bird`.

Model	Scaling	Accuracy	ECE	AdaECE
		CIFAR10-C		
CE	-	75.07 ± 1.4	15.70 ± 1.1	15.68 ± 1.1
CE	TS	75.07 ± 1.4	**12.19 ± 0.9**	**12.17 ± 0.9**
CE	AdaTS	75.07 ± 1.4	**12.03 ± 1.3**	**12.02 ± 1.3**
Brier	-	75.27 ± 0.7	16.21 ± 0.8	16.45 ± 0.7
Brier	TS	75.27 ± 0.7	15.87 ± 0.4	15.86 ± 0.4
Brier	AdaTS	75.27 ± 0.7	**14.84 ± 0.8**	**14.81 ± 0.9**
Ensm	-	77.28	13.45	13.43
Ensm	TS	77.28	10.12	10.09
Ensm	AdaTS	77.21	**9.29**	**9.25**
		CIFAR100-C		
CE	-	51.74 ± 0.4	18.63 ± 0.7	18.58 ± 0.7
CE	TS	51.74 ± 0.4	12.28 ± 1.1	**12.25 ± 1.1**
CE	AdaTS	51.74 ± 0.4	**12.17** ± 0.1	**12.15 ± 0.1**
Brier	-	50.58 ± 0.2	15.04 ± 1.3	15.02 ± 1.3
Brier	TS	50.58 ± 0.2	**9.81 ± 0.8**	**9.81 ± 0.8**
Brier	AdaTS	50.58 ± 0.2	**9.56 ± 0.8**	**9.64 ± 0.7**
Ensm	-	54.61	14.81	14.78
Ensm	TS	54.61	12.66	12.62
Ensm	AdaTS	54.61	**12.02**	**12.00**

Table 2: Corrupted calibration results. Here we can see that adaptive temperate scaling is able to improve calibration on a variety of models. Bold indicates best results, or within one standard deviation of best results.

Misclassification Rejection

Calibrated uncertainty estimates should render that the models are able to reject samples in order to preserve the accuracy. In this setting we report results for AURRA, which

[6]assuming features of clean and corrupted inputs are not mapped exactly to the same point in the embedding space

Methods	AURRA-C	AURRA-DS	AURRA-E
	CIFAR-100		
None	93.07 ± 4.28	91.84 ± 5.24	92.95 ± 4.23
Vanilla TS	92.97 ± 4.25	91.70 ± 5.40	92.67 ± 4.41
Adaptive TS	**93.20 ± 4.25**	**92.00 ± 5.39**	**92.99 ± 4.35**
	Tiny-ImageNet		
None	84.21 ± 1.09	81.83 ± 0.64	83.69 ± 1.16
Vanilla TS	84.05 ± 1.09	81.63 ± 0.64	83.31 ± 1.11
Adaptive TS	**84.68 ± 0.18**	**81.93 ± 0.28**	**84.09 ± 0.20**

Table 3: AURRA scores for based on: confidence (AURRA-C), Demster-Schafer (Sensoy, Kaplan, and Kandemir 2018) (AURRA-DS) and entropy (AURRA-E). Unlike temperature scaling, adaptive temperature scaling does not suffer a reduction in rejection ability. Higher is better.

Methods	Accuracy (↑)	ECE (↓)	AdaECE (↓)
None	85.86 ± 2.48	8.35 ± 1.58	8.15 ± 1.68
Vanilla TS	85.86 ± 2.48	4.57 ± 0.32	4.24 ± 0.43
Adptive TS	85.86 ± 2.48	**3.67 ± 1.41**	**3.35 ± 1.39**

Table 4: CIFAR-10.1, here we see that adaptive temperature scaling is able to provide slightly improved calibration on the harder CIFAR-10.1 dataset.

computes the area under the rejection ratio curve (Nadeem, Zucker, and Hanczar 2009). We display the results in Tab. 3, where we see that adaptive temperature scaling provides a slight improvement over normal predictions and vanilla temperature scaling. Furthermore, we would like to highlight that even though vanilla temperature scaling improves calibration, it does so at the expense of being able to reject samples; unlike adaptive temperature scaling which is able to provide the best of both worlds. It is important to stress that this is a significant advantage, as we are able to provide better calibrated predictions whilst also increasing the models ability to reject samples.

Evaluating Hardness

Given our models ability to predict the temperature, it should naturally extract a notion of hardness, that is how difficult it is to classify. One would expect hard samples to have a high temperature and easy ones to have a low temperature. To conduct this experiment, we utilise the CIFAR-10.1 (Recht et al. 2018) datset, which contains "harder", but statistically similar images to CIFAR-10; conseqently this experiment is not examining data-shift, but is instead measuring the performance on challenging samples. We report the standard metrics: accuracy, ECE and AdaECE in Tab. 4, where we see that adaptive temperature is able to obtain a lower calibration error than vanilla temperature scaling when the model is trained using cross entropy loss.

A key hypothesis we wish to test is "does the model assign higher temperatures to harder samples?"; harder samples should naturally contain a greater amount of uncertainty in their predictions. Consequently, we should see higher temperature values assigned to harder samples (CIFAR-10.1) than to easier ones (CIFAR-10). We test this hypothesis by plotting the histogram of temperature values for CIFAR-10 and CIFAR-10.1, for both correct and incorrect predictions in Fig. 7 (Right).

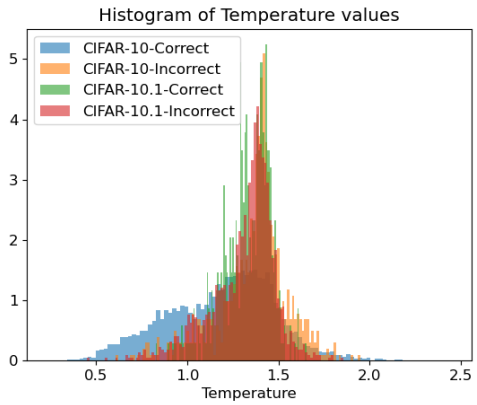

Figure 7: Histograms of temperature for correct predictions for CIFAR-10 and CIFAR-10.1. Lower temperatures are typically assigned to correct (blue) samples from CIFAR-10 but higher for incorrect samples (orange). We also see that hard samples are assigned higher values, regardless of whether they are correct or not (red and green) for CIFAR-10.1.

Here we see that generally, correct samples for CIFAR-10 (blue) are assigned a lower temperature than for CIFAR-10.1 (green), indicating that the adaptive temperature is able to recognise harder samples and assign a higher temperature increasing the uncertainty. We also see that adaptive temperature predicts higher temperatures for incorrect predictions on CIFAR-10 (orange), highlighting adaptive temperatures ability to reduce the confidence of samples which are likely to be incorrect. The same is not true for CIFAR-10.1, this is due to the fact that the samples from CIFAR-10.1 are by design harder, adaptive temperature predicts higher values of T than for the easier CIFAR10 counterpart.

6 Discussion

Here we have presented Adaptive Temperature Scaling (AdaTS), a novel post-hoc method for predicting the temperature on per-sample basis. Given a data-point, our method can predict how confident the classifier should be about its prediction, providing the critical flexibility to increase of decrease the confidence, leading to improved calibration error. The adaptive temperature is also able to obtain better results under distribution shifts. This is achieved by leveraging the latent space of a VAE, which we found to naturally encapsulate and structure the information relating to confidence appropriately. As the model is applied post-hoc, training is very fast, requiring little computational overhead, furthermore it is very easy to implement.

References

Brier, G. W.; et al. 1950. Verification of forecasts expressed in terms of probability. *Monthly weather review*, 78(1): 1–3.

Ding, Y.; Liu, J.; Xiong, J.; and Shi, Y. 2020. Revisiting the evaluation of uncertainty estimation and its application to explore model complexity-uncertainty trade-off. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 4–5.

Guo, C.; Pleiss, G.; Sun, Y.; and Weinberger, K. Q. 2017. On calibration of modern neural networks. In *International Conference on Machine Learning*, 1321–1330. PMLR.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Hendrycks, D.; and Dietterich, T. 2019. Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261*.

Hinton, G. E.; and Zemel, R. S. 1994. Autoencoders, minimum description length and Helmholtz free energy. In *Advances in neural information processing systems*, 3–10.

Kingma, D. P.; and Welling, M. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Krizhevsky, A.; Hinton, G.; et al. 2009. Learning multiple layers of features from tiny images. https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf. Accessed: 2023-05-02.

Lakshminarayanan, B.; Pritzel, A.; and Blundell, C. 2016. Simple and scalable predictive uncertainty estimation using deep ensembles. *Neural Information Processing System*.

Mukhoti, J.; Kulharia, V.; Sanyal, A.; Golodetz, S.; Torr, P. H.; and Dokania, P. K. 2020. Calibrating Deep Neural Networks using Focal Loss. In *NeurIPS*.

Nadeem, M. S. A.; Zucker, J.-D.; and Hanczar, B. 2009. Accuracy-rejection curves (ARCs) for comparing classification methods with a reject option. In *Machine Learning in Systems Biology*, 65–81. PMLR.

Nalisnick, E.; Matsukawa, A.; Teh, Y. W.; Gorur, D.; and Lakshminarayanan, B. 2019. Do Deep Generative Models Know What They Don't Know? In *International Conference on Learning Representations*.

Nixon, J.; Dusenberry, M. W.; Zhang, L.; Jerfel, G.; and Tran, D. 2019. Measuring Calibration in Deep Learning. In *CVPR workshops*, volume 2.

Pinto, F.; Yang, H.; Lim, S.-N.; Torr, P. H.; and Dokania, P. K. 2021. Mix-MaxEnt: improving accuracy and uncertainty estimates of deterministic neural networks. In *NeurIPS Workshop, Distribution shifts: connecting methods and applications*.

Pinto, F.; Yang, H.; Lim, S.-N.; Torr, P. H.; and Dokania, P. K. 2022. RegMixup: Mixup as a Regularizer Can Surprisingly Improve Accuracy and Out Distribution Robustness. In *arXiv: 2206.14502*.

Recht, B.; Roelofs, R.; Schmidt, L.; and Shankar, V. 2018. Do CIFAR-10 Classifiers Generalize to CIFAR-10? https://arxiv.org/abs/1806.00451.

Roelofs, R.; Cain, N.; Shlens, J.; and Mozer, M. C. 2022. Mitigating bias in calibration error estimation. In *International Conference on Artificial Intelligence and Statistics*, 4036–4054. PMLR.

Sensoy, M.; Kaplan, L.; and Kandemir, M. 2018. Evidential deep learning to quantify classification uncertainty. *Advances in Neural Information Processing Systems*, 31.

Tomczak, J.; and Welling, M. 2018. VAE with a VampPrior. In *International Conference on Artificial Intelligence and Statistics*, 1214–1223. PMLR.

Torralba, A.; Fergus, R.; and Freeman, W. T. 2008. 80 Million Tiny Images: A Large Data Set for Nonparametric Object and Scene Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(11): 1958–1970.

Zagoruyko, S.; and Komodakis, N. 2016. Wide residual networks. *arXiv preprint arXiv:1605.07146*.