Trainable Calibration Measures For Neural Networks From Kernel Mean Embeddings

Aviral Kumar [1] Sunita Sarawagi [1] Ujjwal Jain [1]

Abstract

Modern neural networks have recently been found to be poorly calibrated, primarily in the direction of over-confidence. Methods like entropy penalty and temperature smoothing improve calibration by clamping confidence, but in doing so compromise the many legitimately confident predictions. We propose a more principled fix that minimizes an explicit calibration error during training. We present MMCE, a RKHS kernel based measure of calibration that is efficiently trainable alongside the negative likelihood loss without careful hyperparameter tuning. Theoretically too, MMCE is a sound measure of calibration that is minimized at perfect calibration, and whose finite sample estimates are consistent and enjoy fast convergence rates. Extensive experiments on several network architectures demonstrate that MMCE is a fast, stable, and accurate method to minimize calibration error metrics while maximally preserving the number of high confidence predictions.

1. Introduction

Recently, (Guo et al., 2017) made the surprising observation that highly accurate, negative log likelihood trained, deep neural networks predict poorly calibrated confidence probabilities unlike traditional models trained with the same objective (Niculescu-Mizil & Caruana, 2005). Poor calibration implies that if the network makes a prediction with more than 0.99 confidence (which it often does!), the predicted label may be correct much less than 99% of the time. Such lack of calibration is a serious problem in applications like medical diagnosis (Caruana et al., 2015; Crowson et al., 2016; Jiang et al., 2012), obstacle detection in self-driving vehicles (Bojarski et al., 2016), and other applications where learned models feed into decision systems or are human interpreted. Also calibration is useful for detecting out of sample examples (Hendrycks & Gimpel, 2017; Lee et al., 2018; Liang et al., 2018) and in fairness (Pleiss et al., 2017).

A primary reason for poor calibration of modern neural networks is that due to their high capacity, the negative log likelihood (NLL) overfits without overfitting 0/1 error (Zhang et al., 2017). This manifests as overly confident predictions. Recently (Guo et al., 2017) experimented with several known calibration fixes applied post training, and found a simple temperature scaling of logits to be most effective. A second option is to plan for calibration during training. Pereyra et al. (2017) proposes to add a entropy regularizer to the NLL objective to clamp over-confidence. We show that both temperature scaling and entropy regularization manage to reduce aggregate calibration error but in the process needlessly clamp down legitimate high confidence predictions. A third set of approaches model full prediction uncertainty via variational Bayesian networks (Louizos & Welling, 2017), or their committee counterparts (Lakshminarayanan et al., 2017). But their training is too resource-intensive.

We propose a practical and principled fix by minimizing calibration error during training along with classification error. We depend on the power of RKHS functions induced by a universal kernel to express calibration error as a *tractable* integral probability measure, which we call Maximum Mean Calibration Error (MMCE). This is analogous to the way MMD over RKHS kernels expresses the distance between two probability distributions (Muandet et al., 2017; Li et al., 2015). We show that MMCE is a consistent estimator of calibration and converges at rate $1/\sqrt{m}$ to its expected value. Furthermore, MMCE is easy to optimize in the existing mini-batch gradient descent framework.

Our experiments spanning seven datasets show that training with MMCE achieves significant reduction in calibration error while also providing a modest accuracy increase. MMCE achieves this without throttling high confidence predictions. For example on CIFAR-10, MMCE makes 72% predictions at 99.7% confidence, whereas temperature scaling predicts only 40% at 99.6% and entropy scaling only 7% at 96.9%. This is important in applications like medical diagnosis where only highly confident predictions

[1]Department of Computer Science and Engineering, IIT Bombay, Mumbai, India. Correspondence to: Aviral Kumar <aviralkumar2907@gmail.com>.

Proceedings of the 35^{th} International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018. Copyright 2018 by the author(s).

result in saving the cost of manual screening. This study demonstrates that well-designed training methods can simultaneously improve calibration and accuracy without severely reducing the number of high-confidence predictions.

2. Problem Setup

Our focus is improving the calibration of multi-class classification models. Let $\mathcal{Y} = \{1, 2, \ldots, K\}$ denote the set of class labels and $\mathcal{X}$ denote a space of inputs. Let $N_\theta(y|x)$ denote the probability distribution the neural network predicts on an input $x \in \mathcal{X}$ and θ denote the network parameters. For an instance x_i with correct label y_i, the network predicts label $\hat{y}_i = \text{argmax}_{y \in \mathcal{Y}} N_\theta(y|x_i)$. The prediction gets correctness score $c_i = 1$ if $\hat{y}_i = y_i$ and 0 otherwise and a confidence score $r_i = N_\theta(\hat{y}_i|x_i)$. The model $N_\theta(y|x)$ is well-calibrated over a data distribution $\mathcal{D}$, when over all $(x_i, y_i) \in \mathcal{D}$ and $r_i = \alpha$ the probability that $c_i = 1$ is α. For example, out of a sample from $\mathcal{D}$ if 100 examples are predicted with confidence 0.7, then we expect 70 of these to be correct when $N_\theta(y|x)$ is well-calibrated on $\mathcal{D}$. More formally, we use $P_{\theta,\mathcal{D}}(r, c)$ to denote the distribution over r and c values of the predictions of $N_\theta(y|x)$ on $\mathcal{D}$. When $N_\theta(y|x)$ is well calibrated on data distribution $\mathcal{D}$,

$$P_{\theta,\mathcal{D}}(c = 1|r = I_\alpha) = \alpha \quad \forall \alpha \in [0, 1] \tag{1}$$

where I_α denotes a small non-zero interval around α. Using this we can define an expected calibration error (ECE) as

$$\text{ECE}(P_{\theta,\mathcal{D}}) = E_{P_{\theta,\mathcal{D}}(r)}\left[|E_{P_{\theta,\mathcal{D}}(c|r)}[c] - r|\right] \tag{2}$$

To estimate ECE on a data sample $D \sim P_{\theta,\mathcal{D}}$ we partition the [0,1] range of r into B equal bins. We then sum up over each bin $B_j = [\frac{j}{B}, \frac{j+1}{B}]$ the difference between the correctness and confidence scores over examples in that bin:

$$\begin{gathered}\text{ECE}(D) = \frac{1}{|D|}\sum_{j=0}^{B-1} |\sum_{i \in D_j} c_i - \sum_{i \in D_j} r_i| \\ s.t.\ \ D_j = \{i \in D, r_i \in [\frac{j}{B}, \frac{j+1}{B}]\}\end{gathered} \tag{3}$$

We are interested in models with low ECE and high accuracy. Note a model that minimizes ECE may not necessarily have high accuracy. For example, a model that always predicts the majority class with confidence equal to the class's prior probability will have ECE 0 but is not accurate.

Fortunately, the negative log-likelihood loss (NLL=$-\sum_{(x,y)\sim\mathcal{D}} \log N_\theta(y|x)$) used for training neural networks optimizes for accuracy and calibration indirectly. NLL is minimized when $N_\theta(y|x)$ matches the true data distribution $\mathcal{D}$, and is therefore trivially well-calibrated (Hastie et al., 2001). Popular classifiers like linear logistic regression and calibration methods like Platt scaling that optimize the NLL objective lead to well-calibrated models (Niculescu-Mizil & Caruana, 2005). Unfortunately on high capacity neural networks, NLL fails to minimize calibration error because of over-fitting.

We explore ways of training so as to directly optimize for calibration alongside NLL. We need to choose a trainable function CE(D, θ) to measure calibration error for joint optimization with NLL as follows:

$$\min_\theta \text{NLL}(D, \theta) + \lambda\,\text{CE}(D, \theta) \tag{4}$$

We cannot use ECE(D) here because it is highly discontinuous in r and consequently on θ. In the next section we propose a new calibration measure called MMCE that is trainable and satisfies other properties of sound measures that we will discuss next in Section 4.

3. A Trainable Calibration Measure from Kernel Embeddings

Our goal is to design a function to serve as an optimizable surrogate for the calibration error. In this section we propose such a measure that is zero if and only if the model is calibrated and whose finite sample estimates are consistent and enjoy fast convergence rates. Further, we show empirically that it can be optimized over the network parameters θ using existing batch stochastic gradient algorithms.

Our approach is based on defining an integral probability measure over functions from a reproducing kernel Hilbert space (RKHS). Such approaches have emerged as a powerful tool in machine learning and have been successfully used in tasks like comparing two distributions (Gretton et al., 2012), (Li et al., 2015), goodness of fit tests, and class ratio estimation (Iyer et al., 2014) (see (Muandet et al., 2017) for a survey). In this paper, we show their usage in defining a tractable measure of calibration error.

Let $\mathcal{H}$ to denote a reproducible kernel Hilbert space (RKHS) induced by a universal kernel $k(.,.)$ and canonical feature map $\phi : [0, 1] \mapsto \mathcal{H}$. We define a measure called maximum mean calibration error (MMCE) over $P_{\theta,\mathcal{D}}(r, c)$ as:

$$\text{MMCE}(P(r, c)) = \left\|E_{(r,c)\sim P}[(c - r)\phi(r)]\right\|_\mathcal{H} \tag{5}$$

where $\|.\|_\mathcal{H}$ denotes norm in the Hilbert space $\mathcal{H}$. For ease of notation, we use $P(r, c)$ for $P_{\theta,\mathcal{D}}(r, c)$. Theorem 1 in Section 4 shows that MMCE is zero if and only if $P(r, c)$ is calibrated over $\mathcal{D}$ provided kernel $k(.,.)$ is universal. The finite sample estimate over a sample $D \sim P$ with $D = \{(r_1, c_1), \ldots (r_m, c_m)\}$ becomes:

$$\text{MMCE}_m(D) = \left\|\sum_{(r_i,c_i)\in D} \frac{(c_i - r_i)\phi(r_i)}{m}\right\|_\mathcal{H} \tag{6}$$

In Theorem 2 in Section 4 we show that the above estimate is consistent and converges at rate $1/\sqrt{m}$ to MMCE($P(r, c)$).

The above can be rewritten in terms of kernels as:

$$\text{MMCE}^2_m(D) = \sum_{i,j \in D} \frac{(c_i - r_i)(c_j - r_j)k(r_i, r_j)}{m^2} \tag{7}$$

3.1. Minimizing MMCE during training

When training θ on a dataset D we minimize a weighted combination of NLL to reduce classification errors and MMCE to reduce calibration errors. Training is done over mini batches of examples D_b. We found that D_b of size ≈ 100 suffices.

$$\min_{\theta} \sum_{(x_i,y_i)\in D_b} \log N_\theta(y_i|x_i) + \lambda_1 \left(\text{MMCE}^2_m(D_b, \theta)\right)^{\frac{1}{2}} \tag{8}$$

MMCE is differentiable in r but not strictly in θ due to the argmax step for computing the predicted label. During backpropagation we pass the gradient through the r terms but not through the prediction step.

One issue with MMCE during training is that the number of incorrect samples (i.e., $c = 0$) is generally smaller than on test data. We obtained better results by re-weighting examples so that correct and incorrect samples have equal weights. If n denotes the number of correct examples in a batch of size m, we assign a weight of $\frac{m}{n}$ to correct and $\frac{m}{m-n}$ to incorrect examples. The weighted estimate becomes

$$\begin{aligned}\text{MMCE}^2_w(D) = &\sum_{c_i=c_j=0} \frac{r_i r_j k(r_i, r_j)}{(m-n)(m-n)} \\ &+ \sum_{c_i=c_j=1} \frac{(1-r_i)(1-r_j)k(r_i, r_j)}{n^2} \\ &- 2\sum_{c_i=1, c_j=0} \frac{(1-r_i)r_j k(r_i, r_j)}{(m-n)n}\end{aligned} \tag{9}$$

In Section 6 we present an empirical comparison of these two forms of MMCE.

3.2. Why Does MMCE work?

Theoretically, MMCE goes to perfect 0 only at perfect calibration, and minimizing it therefore is sensible. We further try to explain why MMCE works in practice. Note from Eq 6 that MMCE attempts confidence calibration by comparing pairs of instances whereas NLL works on instances individually. To understand why this could prevent overconfident predictions, consider the last term in Eq 9. Now, consider a simple example. Say, in a batch we have an instance x that is misclassified with a high confidence 0.99 and all other correctly classified examples $\mathcal{X}$ with confidence ~ 1. The third term in Eq 9 will pair up x with these high confidence correctly classified examples $\mathcal{X}$. This will exert a *downward pressure* on $\mathcal{X}$s confidence to make it less than 1. In contrast, NLL will continue to push the confidence of $\mathcal{X}$ up towards 1 to the point of over-fitting. We present empirical evidence to support that MMCE does indeed prevent overfitting of NLL. We show test-NLL for MMCE and the baseline with increasing training epochs in Figure 1. The baseline model overfits on test NLL more easily than the MMCE trained model. MMCE is most effective when a batch has a mix of correct and incorrectly classified examples. If training accuracy is 100%, MMCE aligns perfectly with NLL and cannot prevent over-fitting.

4. Analyzing Properties of MMCE

In this section, we prove theoretical properties of MMCE and its finite sample estimate. First, we show that MMCE(P) is a faithful measure of calibration error that is zero if and only if P is perfectly calibrated. Next, we present large deviation bounds of MMCE with finite samples. Finally, we relate the MMCE to ECE.

4.1. MMCE and Perfect Calibration

Theorem 1. *Let $P(r, c)$ be a probability measure defined on the space $\mathcal{X} = ([0, 1] \times \{0, 1\})$ such that $P(r|c = 1)$ and $P(r|c = 0)$ are Borel probability measures. Let k be a universal kernel The MMCE measure (Equation 5) is 0 if and only if P is perfectly calibrated. This implies that MMCE is a proper scoring rule (Parry et al., 2011).*

Proof. We start by defining an intractable integral probability measure for calibration that is more general than MMCE. Let $\mathcal{C}(r)$ denote the space of all continuous bounded functions over $r \in [0, 1]$. The measure is defined as

$$M(\mathcal{C}, P) = \sup_{f\in\mathcal{C}} E_{(r,c)\sim P(r,c)}[(c - r)f(r)] \tag{10}$$

We first show that for calibrated P, $M(\mathcal{C}, P)$ is zero. RHS of above equation is $\int_{r=0}^{1} f(r)[(1 - r)\,\mathrm{dP}(r, c = 1) - r\,\mathrm{dP}(r, c = 0)]$. This is zero because Eq 1 implies that $P(I_r, c = 1) \cdot (1 - r) = r \cdot P(I_r, c = 0)$ and the integration is just a continuous form of this equality.

We prove the converse to show that when P is not calibrated M is non-zero. For uncalibrated P, Eq 1 will not hold on at least one interval $I \subseteq \mathbb{R}$ of non-zero measure, call it $I_0 = [r_0, r_0 + \delta]$ $(\delta > 0)$. Further, since $P(r|c)$ are Borel measures, we can assume that the function $P(I_0, c = 1) \cdot (1 - r_0) - r_0 \cdot P(I_0, c = 0)$ is right continuous and converges to a $s_0 \neq 0$ as $\delta \to 0$. We can then choose a $f_0(r) \in \mathcal{C}(r)$ to be a positive ramp function that is peaked at the center of I_0 and drops to zero outside it. For this choice $E_P((c - r)f_0(r))$ is guaranteed to be non-zero.

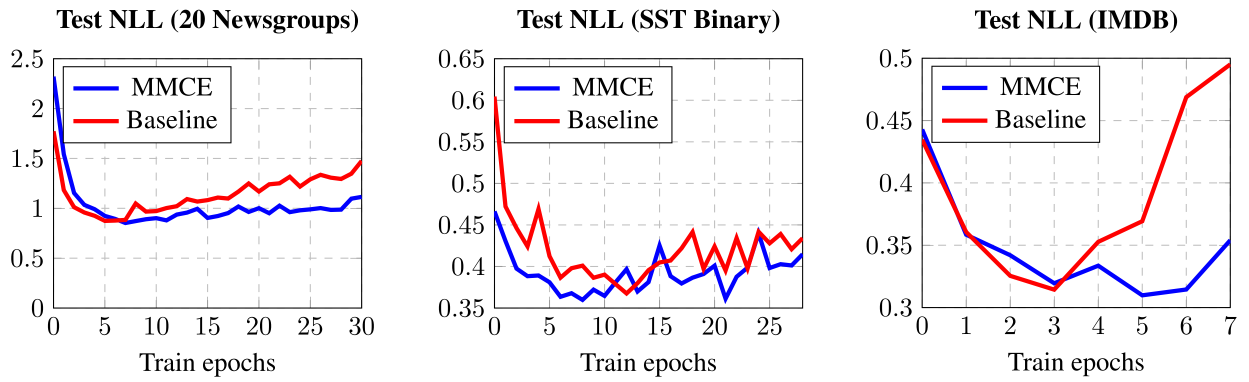

Figure 1. Plots of Test-NLL as the training progresses for 3 datasets/models with MMCE and baseline approaches. Note that the baseline tends to overfit on the train data, whereas MMCE is stable and overfits much less compared to the baseline.

We next proceed to replace the impractical function class $\mathcal{C}$ with functions $\mathcal{F}$ defined over the unit ball in a RKHS space $\mathcal{H}$ with the kernel $k(.,.)$. If k is universal, for every $f \in \mathcal{C}$ and $\epsilon > 0$, there exists a function $f' \in \mathcal{F}$ such that $\max_r |f(r) - f'(r)| < \epsilon$. This property can be combined with above claims to show that $M(\mathcal{F}, P)$ is zero if and only if P is calibrated provided $k(.,.)$ is universal.

Finally we show that $M(\mathcal{F}, P) = \text{MMCE}(P)$

$$\begin{aligned} M(\mathcal{F}, P) &= \sup_{f \in \mathcal{F}} E_{P(r,c)}[(c-r)f(r)] \\ &= \sup_{f \in \mathcal{F}} E_{(r,c)\sim P(r,c)}[(c-r)\langle \phi(r), f\rangle] \\ &= \|E_{r,c\sim P}[(c-r)\phi(r)]\|_{\mathcal{H}} = \text{MMCE}(P) \end{aligned} \tag{11}$$

The first and second equalities in the above are due to the feature mapping properties of RKHS functions. □

4.2. Large Deviation Bounds of MMCE

Theorem 2. *Let $K = \max_{r\in[0,1]} k(r,r)$ and m be size of sample D. With probability more than $1-\delta$*

$$|MMCE_m(D) - MMCE(P)| < \sqrt{\frac{K}{m}}\left(4 + \sqrt{-2\log\delta}\right)$$

Proof in Section 1 of the supplementary material.

4.3. Relationship with ECE

Theorem 3. *$MMCE(P(r,c)) \leq K^{\frac{1}{2}} ECE(P(r,c))$ where $K = \max_r k(r,r)$*

Proof in Section 2 of the supplementary material.

5. Related Work

In classical decision theory, calibration of probabilistic predictions has long been studied under the topic of scoring rules (Parry et al., 2011). In machine learning, many different methods have been proposed for enhancing calibration of classifiers including Platt's scaling (Platt, 1999), Isotonic regression (Zadrozny & Elkan, 2002), and Bayesian binning (Naeini et al., 2015), to name a few. A systematic study in (Niculescu-Mizil & Caruana, 2005) found logistic regression classifier and neural networks (of 2005) to be well-calibrated, and the ones that were not calibrated (e.g. SVMs) were best fixed via Platt scaling. Candela et al. (2005) discusses ways of measuring calibration error and highlights the potential of over-fitting of NLL. Kuleshov & Liang (2015) studied calibration for structured prediction and Kuleshov & Ermon (2017) shows how to calibrate in an online adversarial setting.

For modern neural networks, a recent systematic study (Guo et al., 2017) finds them to be surprisingly poorly calibrated. They compare several conventional post-facto fixes and find temperature scaling to provide the best calibration. Another recent work proposes an entropy regularizer during training to penalize over-confident predictions (Pereyra et al., 2017). A shortcoming of both methods is that they indiscriminately clamp the confidence of all predictions, robbing an end application of the benefit of high confidence predictions. Bayesian inference is a principled approach to get better prediction uncertainties (Louizos & Welling, 2017) but incurs significantly higher training cost. Lakshminarayanan et al. (2017) proposes to average predictions from a committee of models but this is resource-intensive.

Our use of kernel embeddings to measure calibration error is similar to their use in MMD to measure distance between distributions (Gretton et al., 2012). MMD has recently been used for training various types of neural networks, including generative models in (Li et al., 2015) and unsupervised domain-adaptation models in (Yan et al., 2017). For other measures defined on kernel embeddings see (Muandet et al., 2017). We are aware of no prior work on defining calibration errors using kernel embeddings.

6. Experiments

We compare our method of calibrating using MMCE with existing calibration methods on seven datasets spanning image, NLP, and time-series and 6 network architectures. We establish that a model trained with MMCE not only makes better calibrated and accurate predictions but does so at higher confidence level than existing approaches. We also evaluate the computational overhead of training with the quadratic MMCE loss, and show that the increase in running time is no more than 10%.[1]

Datasets The datasets used in our experiments are:

1. CIFAR-10 (Krizhevsky et al., a): Color images (32×32) from 10 classes. 45,000/5,000/10,000 images for train/validation/test.

2. CIFAR-100 (Krizhevsky et al., b): Same as above but with 100 classes.

3. Caltech Birds 200 (Welinder et al., 2010): Images of 200 bird species drawn from Imagenet. 5994/2897/2897 images for train/validation/test sets.

4. 20 Newsgroups: News articles partitioned into 20 categories by content. 15098/900/3999 documents for train/validation/test.

5. IMDB reviews (Maas et al., 2011): Polar movie reviews for sentiment classification 25000/5000/20000 for train/ validation/ test.

6. UC Irvine Human Activity Recognition(HAR) (Anguita et al., 2013): Time series from phones corresponding to 6 human actions. 6653/699/2947 instances for train/ validation/ test.

7. Stanford Sentiment Treebank (SST) (Socher et al., 2012): Movie reviews, represented as parse trees that are annotated by sentiment. Each sample includes a binary label and a fine grained 5-class label. We used the binary version. Training/validation/test sets contain 6920/872/1821 documents.

Models For the first three image datasets we used state-of-the-art convolutional networks like Resnet (He et al., 2016), Wide Resnet (Zagoruyko & Komodakis, 2016) and Inception v3 (Krause et al., 2016). We use different sizes (in terms of depths/ number of layers) for each of the models. On 20 Newsgroups, we train a global pooling Convolutional Network (Lin et al., 2013). On IMDB reviews, we use a version of hierarchical attention networks (Yang et al., 2016). On UCI HAR, we use a LSTM. We used the TreeLSTM model for SST binary (Tai et al., 2015). We used publicly available models: CIFAR 10 Resnet: (Tensorflow, 2018), CIFAR 10 wide resnet, CIFAR 100 all models: (Xin Pan, 2018), Birds CUB dataset: (Vispedia, 2018), IMDB dataset: (Ilya Ivanov, 2018), 20 Newsgroups: (Keras Team, 2018), HAR dataset: (Guillaume Chevalier, 2017) and SST TreeLSTM : (Nicolas Pinchaud, 2017).

[1]Code is partially available and will be made fully available at https://github.com/aviralkumar2907/MMCE.

Experiment setup The λ for weighting MMCE wrt NLL is chosen via cross-validation. The same kernel $k(r, r')$ was used for all since r, r' are probabilities. We chose the Laplacian Kernel $k(r, r') = \exp\left(\frac{-|r-r'|}{0.4}\right)$, a universal kernel, with a width of 0.4. For measuring calibration error we use ECE with 20 bins, each of size 0.05. We use a batch size of 128, except when the default batch size in the code base was higher. For example, in the IMDB HAN codebase, the default batch size was 256 and for UCI HAR the batch size was 1500. Other details about optimizer and hyper-parameters were kept unchanged from the base models from the downloaded source. As an exception, the batch size for SST was kept fixed to 25 (the default value).

Methods Compared We compare the following methods:

Baseline The baseline models in all tasks were trained using negative log likelihood. All baseline models were downloaded from public sources and came with their best performing regularizers like dropout and weight penalty.

Baseline+T Guo et al. (2017) compared six methods of calibrating a baseline model using a validation dataset. Of these temperature scaling was found to be the best in terms of ECE and the only method to not drop baseline accuracy. We compare with this approach and use Baseline+T to denote a temperature calibrated baseline model.

MMCE, MMCE$_m$, MMCE+T These are variants of our MMCE-based approach. MMCE refers to the weighted version MMCE$_w$ in Eq 9 and MMCE$_m$ to unweighted version in Eq 7. Temperature scaling can also be applied on models trained using MMCE. We call this the MMCE+T method.

Entropy penalty We also compare with (Pereyra et al., 2017) that adds an entropy penalty as a regularizer to reduce over-confidence, a primary cause of poor calibration in modern NN.

Kernel regression MMCE uses kernels to measure calibration. Another conventional kernel-based measure is the Nardaya-Watson kernel regressor. We use this to get a smoothed estimate of $P(c = 1|r)$ and then minimize their distance from r to achieve calibration as per Eq 1. This gives us a trainable measure of calibration error as

$$\sum_{i \in D} \left(r_i - \frac{\sum_{j \in D} k(r_i, r_j) c_j}{\sum_{j \in D} k(r_i, r_j)} \right)^2. \tag{12}$$

that we use as CE(D, θ) in Equation 4.

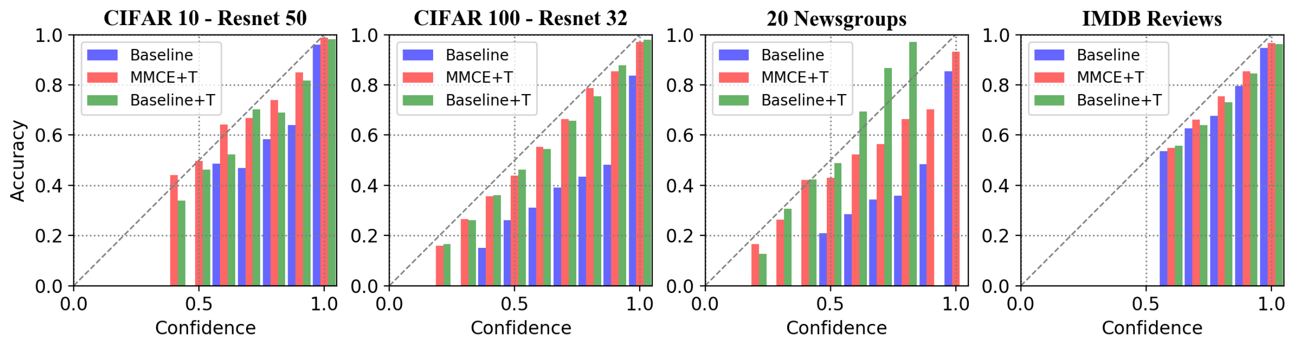

Figure 2. Reliability diagram to show calibration of Baseline, MMCE+T, and Baseline+T. The x-axis is prediction confidence in ten bins of equal size and y-axis is accuracy at that bin's confidence. Perfect measure should touch the line. While plotting, we dropped bins with less than 0.5% of the total data points falling into them.

Evaluation Metrics We compare the methods on three different measures of calibration error (Candela et al., 2005) - Expected Calibration Error (ECE) (as described in Section 2), Brier Score ($\frac{1}{|\mathcal{D}|}\sum_{(x_i,y_i)\sim\mathcal{D}}\sum_v(c_{i,v}-r_{i,v})^2$), and NLL. We also measure the fraction of highly-confident calibrated predictions to reward methods that achieve calibration without throttling correct high confidence scores.

6.1. Results

MMCE gains over baseline We first evaluate the efficacy of MMCE in reducing calibration errors over the baseline without dropping accuracy. Tables 1 and Table 2 summarizes the results over different dataset architecture combinations. From columns 3 and 4 of Table 1 we observe that MMCE training consistently gives lower ECE values. For example, on CIFAR-10 all three models get almost a factor of 4 reduction in ECE. The reduction in calibration error is also evident when measured on Brier score and NLL(Table 2). MMCE also boosts baseline accuracy showing that accuracy (as optimized by NLL) is not orthogonal to calibration (as optimized by MMCE).

All the models here were trained from scratch with MMCE. But we show that MMCE can be used also for fine-tuning a pre-trained model. Table 1 in the supplementary material shows the ECE and accuracy comparisons with a baseline model fine-tuned using our NLL+λMMCE objective. We find that fine-tuning with MMCE helps improve calibration, although not as much as training from scratch.

Comparison with temperature scaling In the last two columns of Table 1 we compare the effect on ECE of temperature scaling (TS) on both the baseline and MMCE trained model. Accuracy stays unchanged with TS. We find that both models get an improvement in calibration (drop in ECE) with TS. Calibration is best overall with MMCE+T. On 20newsgroup, MMCE+T provides a particularly big improvement over Baseline+T. From Table 2 we observe that on Brier and NLL too, MMCE (even without TS) provides greater reduction in calibration error over baseline+TS. One natural question is why should temperature scaling improve MMCE? The reason we suspect is the injection of a different validation dataset. TS using MMCE (instead of NLL) on the validation dataset gives similar drops in ECE. But, MMCE gives a much greater fraction of the increase beyond baseline than TS. 70% of the ECE reduction of MMCE+T is due to MMCE training and only additional 30% with TS on the validation dataset.

We zoom beyond aggregated ECE values and compare Baseline, Baseline+T, and MMCE+T on reliability plots in Figure 2. In a reliability plot, the perfectly calibrated model should touch the diagonal, models below the diagonal are over-confident and above it are under-estimating confidence. The Baseline is overly confident and temperature scaling (TS) corrects that significantly. But, on 20newsgroup, TS causes the Baseline model to swing in the opposite direction of under confidence!

We further show that TS incurs confidence loss on several other datasets but the loss happens at the very high end and does not get reflected in aggregate ECE numbers. Since over-confidence is the problem, we focus on the subset of predictions made with very high confidence, specifically 0.99. For each method, we define a confident set (CS99) consisting of predictions with confidence above 0.99. We desire the largest possible set CS99 as long as accuracy in CS99 is above 99%. We show the result in Table 4. The over-confident baseline, of course, gets the largest size of CS99 but its accuracy is below the contract of 99% on all five cases. Temperature scaling on the baseline does correct accuracy but the size of the confident set is *much* smaller now. In contrast MMCE (both with and without TS) provides much larger number of confident predictions. For example, on CIFAR 10 (Resnet 110), temperature scaling reduced the size of CS99 from 89% to 40% whereas MMCE lifted it to

E#	Dataset	Model	ECE		Accuracy		ECE with Temperature	
			Baseline	MMCE	Baseline	MMCE	Baseline+T	MMCE+T
1	MNIST	LeNet 5	0.5%	**0.2%**	99.24%	99.26%	–	–
2	CIFAR 10	Resnet 50	4.3%	**1.2%**	93.1%	93.4%	**0.9%**	1.1%
3	CIFAR 10	Resnet 110	4.6%	**1.1%**	93.7%	94.0%	1.21%	**1.19%**
4	CIFAR 10	Wide Resnet 28-10	4.5%	**1.6%**	94.1%	94.2%	**1.0%**	1.1%
5	CIFAR 100	Resnet 32	19.6%	**6.9%**	67.0%	67.7%	2.5%	**1.4%**
6	CIFAR 100	Wide Resnet 28-10	15.0%	**8.9%**	74.0%	76.6%	2.5%	**2.3%**
7	Birds CUB 200	Inception-v3	2.6%	**2.3%**	78.2%	77.9%	2.0%	**1.4%**
8	20 Newsgroups	Global Pooling CNN	16.5%	**6.5%**	74.2%	73.9%	15.8%	**6.2%**
9	IMDB Reviews	HAN	4.9%	**0.4%**	86.8%	86.3%	1.0%	**0.2%**
10	SST Binary	Tree LSTM	7.4%	**5.9%**	88.6%	88.7%	**1.8%**	3.6%
11	HAR time series	LSTM	7.6%	**5.9%**	89.4%	90.3%	3.8%	**3.7%**

Table 1. ECE and Accuracy for Baseline and MMCE after training from scratch and after temperature scaling using a validation dataset. Here a model is referenced by the name and the size of its network.

E#	Brier Score			Test NLL		
	Base	MMCE	B+T	Base	MMCE	B+T
2	0.121	**0.090**	0.111	0.31	**0.21**	0.23
4	0.087	0.085	**0.082**	0.24	0.18	0.18
5	0.522	**0.445**	0.467	1.74	**1.24**	1.26
6	0.340	**0.338**	**0.338**	0.96	0.94	**0.93**
7	0.318	**0.312**	0.317	0.87	**0.86**	**0.86**
8	0.401	**0.365**	0.380	1.34	**0.97**	1.02
9	0.197	**0.189**	0.190	0.33	**0.30**	0.31
10	0.220	**0.209**	0.218	0.42	**0.40**	0.41
11	0.183	**0.172**	0.178	0.49	**0.34**	0.38

Table 2. Brier Score and NLL Values for Baseline, MMCE and Baseline+T (B+T) after training from scratch and after temperature scaling using a validation dataset. Experiment numbers (E#) can be looked up from Table 1

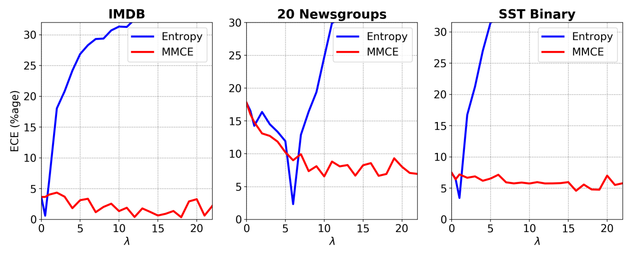

Figure 3. Variation of ECE with regularizer weight λ with Entropy penalty and MMCE as regularizers. Observe the sharp peaks of Entropy penalty as against the stable trends of MMCE. The values on the y-axis were clipped to ~ 30 for visual clarity.

72% without dropping accuracy. As pointed out earlier, less (or no) highly confident predictions is not good from the perspective of practical deployment of neural networks.

Some applications rely on highly confident predictions, for example, applications of neural networks in medicine and healthcare. The ideal goal of confidence calibration must be to find not only accurate but highly confident predictions. Clamping down large confidence values is not the best way to tap the potential of modern high capacity networks and we need training time methods like MMCE to wean out incorrect confident predictions.

Comparison with Alternative Regularizers We next compare MMCE with Entropy penalty and find that indeed this method reduces ECE significantly even though the goal was to fix over-confidence and not calibration. However, the method suffers from the same flaw as temperature scaling of indiscriminately clamping confidence leading to only a few confident predictions as seen in Table 4. For example, on CIFAR 10 with a threshold of 0.99 on confidence, MMCE predicts 72% instances at accuracy 99.7% but with Entropy penalty we get only 7% and that too at 97% accuracy indicating miscalibration in the above 0.99 bucket.

The Kernel regression regularizer (Eq 12) was not effective in improving calibration. This was surprising because the Nardaya-Watson regressor has a better convergence rate $m^{-\frac{4}{5}}$ than ours $m^{-\frac{1}{2}}$. A possible reason could be that the gradient with respect to θ is not conducive to optimizing the objective, perhaps because of kernels in the denominator. In contrast, MMCE with kernels only in the numerator is conducive to gradient-based optimization.

MMCE: weighted vs unweighted In Section 3, Equation 7 we motivated the design of the weighted MMCE estimator to surmount natural high accuracy bias in the training data. In Table 3 we observe that this version (last column) indeed provides a significant boost in accuracy over the original MMCE version on most cases.

Dataset	Model	MMCE_m		Kernel Regression		Entropy penalty		MMCE	
		ECE	Acc%	ECE	Acc%	ECE	Acc%	ECE	Acc%
CIFAR 10	Resnet 50	3.2%	93.2%	4.9%	93.1%	1.8%	93.0%	**1.2%**	93.4%
CIFAR 10	Resnet 110	4.1%	93.6%	4.5%	93.6%	2.1%	93.8%	**1.2%**	94.0%
CIFAR 10	Wide Resnet 28-10	3.0%	95.0%	3.1%	95.1%	2.8%	94.7%	**1.6%**	94.2%
CIFAR 100	Wide Resnet 28-10	10.4%	77.1%	12.2%	78.6%	9.3%	77.9%	**8.9%**	76.6%
HAR time series	LSTM	5.5%	89.8%	7.0%	90.2%	**2.7%**	91.8%	5.9%	90.3%
20 Newsgroups	Global Pool CNN	15.8%	75.1%	18.8%	75.1%	**2.3%**	74.2%	6.5%	73.9%
IMDB Reviews	HAN	3.6%	86.6%	3.2%	85.7%	0.6%	86.6%	**0.4%**	86.3%

Table 3. ECE and accuracy over different methods that train using NLL+λ CE. Observe that entropy penalty reduces ECE without dropping accuracy. Kernel regression helps only in some cases and weighted MMCE is much better than the unweighted one (MMCE_m).

Dataset	Model	Baseline		Baseline + T		MMCE		MMCE + T		Entropy	
		|CS99|	Acc.	|CS99|	Acc.	|CS99|	Acc.	|CS99|	Acc.	|CS99|	Acc.
CIFAR 10	Resnet 110	89	97.65	40	99.57	72	99.69	68	99.79	7	96.96
CIFAR 10	Resnet 50	86	98.04	51	99.68	67	99.68	66	99.71	26	98.04
IMDB	HAN	48	98.28	12	99.48	19.3	99.40	19.1	99.24	18.6	99.20
20 NewsG	Global Pool	61	92.07	0	–	45	96.26	42	96.73	7	97.14
Birds	Inception v3	55	98.68	40	99.31	48	99.34	41	99.40	37	99.01

Table 4. The fraction of test examples predicted with a confidence more than 99% and their accuracy. Observe that fixes like temperature scaling and entropy penalty move many more points out of the high confidence zone than MMCE.

Tuning Calibration Weight λ In Figure 3 we show the effect on ECE for different values of λ in the NLL+λ CE training objective for MMCE and Entropy regularizer as CE. We find that compared to Entropy, MMCE is significantly more stable with changing lambda. For example for 20Newsgroup, the entropy regularizer has a minima at 6 and rises sharply on both sides whereas MMCE moves gradually towards its flat minima. The accuracy values too (not shown in Figure) do not change much with λ for MMCE.

Computational Efficiency We compared the running times per epoch of the baseline and the MMCE trained model (on a NVIDIA Titan X GPU) (In Table 2 of supplementary). Although, MMCE scales quadratically with the number of examples we found that the wall-clock running time per epoch for the MMCE algorithm was no more than 10% over the baseline. This is because the MMCE is reliably estimated even on ~ 100 examples that comprise a batch because of its fast convergence properties.

In summary, our experiments demonstrate that training with MMCE is a fast, stable, and accurate method to minimize calibration error while maximally preserving high confidence predictions.

7. Conclusion

We proposed MMCE a new measure of calibration error in terms of universal kernels from a RKHS space. We analyzed MMCE theoretically and showed that it is minimized only when the model is calibrated and its finite sample estimate is consistent and converges at rate $1/\sqrt{m}$. The measure can be used to jointly train the network parameters to minimize both accuracy and calibration error. The joint objective is easy to train and does not require very careful hyper-parameter tuning unlike other similar regularizers e.g. the entropy regularizer. Also, the running time overhead is minimal unlike methods like Bayesian networks which model prediction uncertainty at huge performance penalty. Our model not only provides low ECE and high accuracy, but also produces more predictions at high confidence levels unlike previous smoothing approaches that indiscriminately clamp large confidence scores. Our experiments showed that modern neural networks suffer more from poor calibration than over-confidence. Careful training objectives, like MMCE, can fix poor calibration without reducing confidence.

In future, we plan to apply MMCE to calibrate structured prediction tasks, for example, the sequence to sequence models used for translation.

Acknowledgements We thank all anonymous reviewers for their comments and for pointing to the work on scoring rules in statistics. We thank members of our group at IIT Bombay, specifically Shiv Shankar, for discussions. We gratefully acknowledge the support of NVIDIA corporation for Titan X GPUs.

References

Anguita, D., Ghio, A., Oneto, L., Parra, X., and L Reyes-Ortiz, J. A public domain dataset for human activity recognition using smartphones, 01 2013.

Bojarski, M., Testa, D. D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L. D., Monfort, M., Muller, U., Zhang, J., Zhang, X., Zhao, J., and Zieba, K. End to end learning for self-driving cars. *CoRR*, abs/1604.07316, 2016. URL http://arxiv.org/abs/1604.07316.

Candela, J. Q., Rasmussen, C. E., Sinz, F. H., Bousquet, O., and Schölkopf, B. Evaluating predictive uncertainty challenge. In *Machine Learning Challenges, Evaluating Predictive Uncertainty, Visual Object Classification and Recognizing Textual Entailment, First PASCAL Machine Learning Challenges Workshop, MLCW 2005, Southampton, UK, April 11-13, 2005, Revised Selected Papers*, pp. 1–27, 2005.

Caruana, R., Lou, Y., Gehrke, J., Koch, P., Sturm, M., and Elhadad, N. Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. In *Proceedings of the 21th KDD 2015*, KDD '15, pp. 1721–1730, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3664-2. doi: 10.1145/2783258.2788613.

Crowson, C. S., Atkinson, E. J., and Therneau, T. M. Assessing calibration of prognostic risk scores. *Statistical Methods in Medical Research*, 25(4):1692–1706, 2016.

Gretton, A., Borgwardt, K. M., Rasch, M. J., Schölkopf, B., and Smola, A. J. A kernel two-sample test. *Journal of Machine Learning Research*, 13:723–773, 2012.

Guillaume Chevalier. LSTMs for Human Activity Recognition. https://github.com/guillaume-chevalier/LSTM-Human-Activity-Recognition, 2017. Online; accessed 27 January 2018.

Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pp. 1321–1330, 2017.

Hastie, T., Tibshirani, R., and Friedman, J. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. *CVPR*, 2016.

Hendrycks, D. and Gimpel, K. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *ICLR*, 2017.

Ilya Ivanov. IMDB HAN model. https://github.com/ilivans/tf-rnn-attention, 2018. Online; accessed 27 January 2018.

Iyer, A., Nath, S., and Sarawagi, S. Maximum mean discrepancy for class ratio estimation: Convergence bounds and kernel selection. In *ICML*, 2014.

Jiang, X., Osl, M., Kim, J., and Ohno-Machado, L. Calibrating predictive model estimates to support personalized medicine. *Journal of the American Medical Informatics Association*, 19(2):263–274, 2012. doi: 10.1136/amiajnl-2011-000291.

Keras Team. Keras pre-trained word embeddings example. https://github.com/keras-team/keras/blob/master/examples/pretrained_word_embeddings.py, 2018. Online; accessed 24 January 2018.

Krause, J., Sapp, B., Howard, A., Zhou, H., Toshev, A., Duerig, T., Philbin, J., and Fei-Fei, L. The unreasonable effectiveness of noisy data for fine-grained recognition. In Leibe, B., Matas, J., Sebe, N., and Welling, M. (eds.), *Computer Vision – ECCV 2016*, pp. 301–320, Cham, 2016. Springer International Publishing.

Krizhevsky, A., Nair, V., and Hinton, G. Cifar-10 (canadian institute for advanced research). a. URL http://www.cs.toronto.edu/~kriz/cifar.html.

Krizhevsky, A., Nair, V., and Hinton, G. Cifar-100 (canadian institute for advanced research). b. URL http://www.cs.toronto.edu/~kriz/cifar.html.

Kuleshov, V. and Ermon, S. Estimating uncertainty online against an adversary. In *AAAI*, 2017.

Kuleshov, V. and Liang, P. S. Calibrated structured prediction. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R. (eds.), *NIPS*, pp. 3474–3482. 2015.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. Simple and scalable predictive uncertainty estimation using deep ensembles. In *NIPS*, pp. 6405–6416. 2017.

Lee, K., Lee, H., Lee, K., and Shin, J. Training confidence-calibrated classifiers for detecting out-of-distribution samples. In *ICLR*, 2018.

Li, Y., Swersky, K., and Zemel, R. Generative moment matching networks. In Bach, F. and Blei, D. (eds.), *ICML*, volume 37 of *PMLR*, pp. 1718–1727, Lille, France, 07–09 Jul 2015. PMLR.

Liang, S., Li, Y., and Srikant, R. Enhancing the reliability of out-of-distribution image detection in neural networks. *ICLR*, 2018.

Lin, M., Chen, Q., and Yan, S. Network in network. *CoRR*, abs/1312.4400, 2013.

Louizos, C. and Welling, M. Multiplicative normalizing flows for variational Bayesian neural networks. In *ICML*, volume 70, pp. 2218–2227, 2017.

Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., and Potts, C. Learning word vectors for sentiment analysis. In *Proceedings of ACL: HLT*, June 2011.

Muandet, K., Fukumizu, K., Sriperumbudur, B. K., and Schölkopf, B. Kernel mean embedding of distributions: A review and beyond. *Foundations and Trends in Machine Learning*, 10(1-2):1–141, 2017.

Naeini, M. P., Cooper, G. F., and Hauskrecht, M. Obtaining well calibrated probabilities using bayesian binning. In *AAAI*, 2015.

Nicolas Pinchaud. Fast Minibatch version of Tree LSTM. https://github.com/nicolaspi/treelstm, 2017. Online; accessed 31 January 2018.

Niculescu-Mizil, A. and Caruana, R. Predicting good probabilities with supervised learning. In *ICML*, 2005.

Parry, M., Dawid, A. P., and Lauritzen, S. Proper local scoring rules. *ArXiv e-prints*, January 2011.

Pereyra, G., Tucker, G., Chorowski, J., Kaiser, L., and Hinton, G. E. Regularizing neural networks by penalizing confident output distributions. *ICLR workshop*, 2017.

Platt, J. C. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *ADVANCES IN LARGE MARGIN CLASSIFIERS*, pp. 61–74. MIT Press, 1999.

Pleiss, G., Raghavan, M., Wu, F., Kleinberg, J. M., and Weinberger, K. Q. On fairness and calibration. In *NIPS*, pp. 5684–5693, 2017.

Socher, R., Huval, B., Manning, C. D., and Ng, A. Y. Semantic Compositionality Through Recursive Matrix-Vector Spaces. In *EMNLP*, 2012.

Tai, K. S., Socher, R., and Manning, C. D. Improved semantic representations from tree-structured long short-term memory networks. In *ACL*, 2015.

Tensorflow. CIFAR 10 model. https://github.com/tensorflow/models/tree/master/official/resnet, 2018. Online; accessed 21 January 2018.

Vispedia. Classification - Birds CUB 200 dataset . https://github.com/visipedia/tf_classification, 2018. Online; accessed 26 January 2018.

Welinder, P., Branson, S., Mita, T., Wah, C., Schroff, F., Belongie, S., and Perona, P. Caltech-UCSD Birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology, 2010.

Xin Pan. CIFAR 100 model. https://github.com/tensorflow/models/tree/master/research/resnet, 2018. Online; accessed 21 January 2018.

Yan, H., Ding, Y., Li, P., Wang, Q., Xu, Y., and Zuo, W. Mind the class weight bias: Weighted maximum mean discrepancy for unsupervised domain adaptation. In *2017 IEEE CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pp. 945–954, 2017.

Yang, Z., Yang, D., Dyer, C., He, X., Smola, A. J., and Hovy, E. H. Hierarchical attention networks for document classification. In *HLT-NAACL*, 2016.

Zadrozny, B. and Elkan, C. Transforming classifier scores into accurate multiclass probability estimates. In *ACM SIGKDD*, 2002.

Zagoruyko, S. and Komodakis, N. Wide residual networks. *CoRR*, abs/1605.07146, 2016.

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. Understanding deep learning requires rethinking generalization. *ICLR*, 2017.

$$(1-\alpha)P(c=1, r=\alpha) = \alpha P(c=0, r=\alpha) \quad \forall \alpha \in [0,1]$$

$$\sum_{c \in \{0,1\}} (c-\alpha)P(c, r=\alpha) = 0 \quad \forall \alpha \in [0,1]$$