On Calibration of Modern Neural Networks

Chuan Guo [*1] **Geoff Pleiss** [*1] **Yu Sun** [*1] **Kilian Q. Weinberger** [1]

Abstract

Confidence calibration – the problem of predicting probability estimates representative of the true correctness likelihood – is important for classification models in many applications. We discover that modern neural networks, unlike those from a decade ago, are poorly calibrated. Through extensive experiments, we observe that depth, width, weight decay, and Batch Normalization are important factors influencing calibration. We evaluate the performance of various post-processing calibration methods on state-of-the-art architectures with image and document classification datasets. Our analysis and experiments not only offer insights into neural network learning, but also provide a simple and straightforward recipe for practical settings: on most datasets, *temperature scaling* – a single-parameter variant of Platt Scaling – is surprisingly effective at calibrating predictions.

1. Introduction

Recent advances in deep learning have dramatically improved neural network accuracy (Simonyan & Zisserman, 2015; Srivastava et al., 2015; He et al., 2016; Huang et al., 2016; 2017). As a result, neural networks are now entrusted with making complex decisions in applications, such as object detection (Girshick, 2015), speech recognition (Hannun et al., 2014), and medical diagnosis (Caruana et al., 2015). In these settings, neural networks are an essential component of larger decision making pipelines.

In real-world decision making systems, classification networks must not only be accurate, but also should indicate when they are likely to be incorrect. As an example, consider a self-driving car that uses a neural network to detect pedestrians and other obstructions (Bojarski et al., 2016).

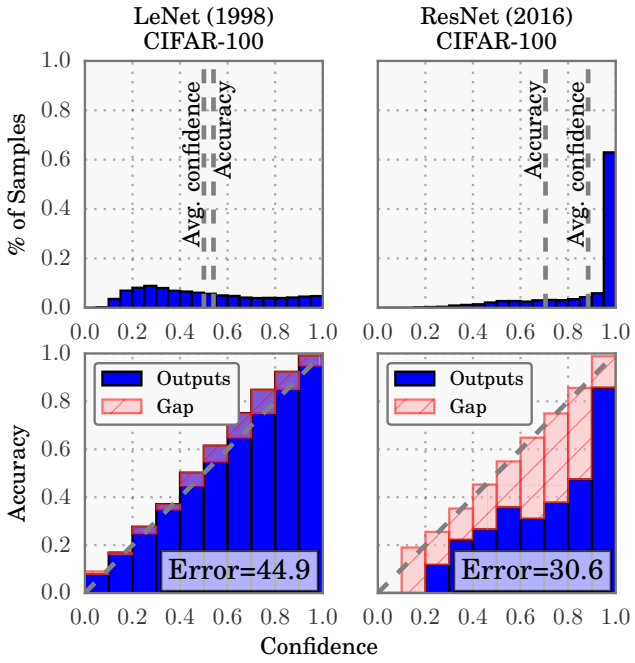

Figure 1. Confidence histograms (top) and reliability diagrams (bottom) for a 5-layer LeNet (left) and a 110-layer ResNet (right) on CIFAR-100. Refer to the text below for detailed illustration.

If the detection network is not able to confidently predict the presence or absence of immediate obstructions, the car should rely more on the output of other sensors for braking. Alternatively, in automated health care, control should be passed on to human doctors when the confidence of a disease diagnosis network is low (Jiang et al., 2012). Specifically, a network should provide a *calibrated confidence* measure in addition to its prediction. In other words, the probability associated with the predicted class label should reflect its ground truth correctness likelihood.

Calibrated confidence estimates are also important for model interpretability. Humans have a natural cognitive intuition for probabilities (Cosmides & Tooby, 1996). Good confidence estimates provide a valuable extra bit of information to establish trustworthiness with the user – especially for neural networks, whose classification decisions are often difficult to interpret. Further, good probability estimates can be used to incorporate neural networks into other probabilistic models. For example, one can improve performance by combining network outputs with a lan-

[*]Equal contribution, alphabetical order. [1]Cornell University. Correspondence to: Chuan Guo <cg563@cornell.edu>, Geoff Pleiss <geoff@cs.cornell.edu>, Yu Sun <ys646@cornell.edu>.

Proceedings of the 34^{th} International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017.

guage model in speech recognition (Hannun et al., 2014; Xiong et al., 2016), or with camera information for object detection (Kendall & Cipolla, 2016).

In 2005, Niculescu-Mizil & Caruana (2005) showed that neural networks typically produce well-calibrated probabilities on binary classification tasks. While neural networks today are undoubtedly more accurate than they were a decade ago, we discover with great surprise that *modern neural networks are no longer well-calibrated*. This is visualized in Figure 1, which compares a 5-layer LeNet (left) (LeCun et al., 1998) with a 110-layer ResNet (right) (He et al., 2016) on the CIFAR-100 dataset. The top row shows the distribution of prediction confidence (i.e. probabilities associated with the predicted label) as histograms. The average confidence of LeNet closely matches its accuracy, while the average confidence of the ResNet is substantially higher than its accuracy. This is further illustrated in the bottom row reliability diagrams (DeGroot & Fienberg, 1983; Niculescu-Mizil & Caruana, 2005), which show accuracy as a function of confidence. We see that LeNet is well-calibrated, as confidence closely approximates the expected accuracy (i.e. the bars align roughly along the diagonal). On the other hand, the ResNet's accuracy is better, but does not match its confidence.

Our goal is not only to understand why neural networks have become miscalibrated, but also to identify what methods can alleviate this problem. In this paper, we demonstrate on several computer vision and NLP tasks that neural networks produce confidences that cannot represent true probabilities. Additionally, we offer insight and intuition into network training and architectural trends that may cause miscalibration. Finally, we compare various post-processing calibration methods on state-of-the-art neural networks, and introduce several extensions of our own. Surprisingly, we find that a single-parameter variant of Platt scaling (Platt et al., 1999) – which we refer to as *temperature scaling* – is often the most effective method at obtaining calibrated probabilities. Because this method is straightforward to implement with existing deep learning frameworks, it can be easily adopted in practical settings.

2. Definitions

The problem we address in this paper is supervised multi-class classification with neural networks. The input $X \in \mathcal{X}$ and label $Y \in \mathcal{Y} = \{1, \dots, K\}$ are random variables that follow a ground truth joint distribution $\pi(X, Y) = \pi(Y|X)\pi(X)$. Let h be a neural network with $h(X) = (\hat{Y}, \hat{P})$, where $\hat{Y}$ is a class prediction and $\hat{P}$ is its associated confidence, i.e. probability of correctness. We would like the confidence estimate $\hat{P}$ to be calibrated, which intuitively means that $\hat{P}$ represents a true probability. For example, given 100 predictions, each with confidence of 0.8, we expect that 80 should be correctly classified. More formally, we define *perfect calibration* as

$$\mathbb{P}\left(\hat{Y} = Y \mid \hat{P} = p\right) = p, \quad \forall p \in [0, 1] \tag{1}$$

where the probability is over the joint distribution. In all practical settings, achieving perfect calibration is impossible. Additionally, the probability in (1) cannot be computed using finitely many samples since $\hat{P}$ is a continuous random variable. This motivates the need for empirical approximations that capture the essence of (1).

Reliability Diagrams (e.g. Figure 1 bottom) are a visual representation of model calibration (DeGroot & Fienberg, 1983; Niculescu-Mizil & Caruana, 2005). These diagrams plot expected sample accuracy as a function of confidence. If the model is perfectly calibrated – i.e. if (1) holds – then the diagram should plot the identity function. Any deviation from a perfect diagonal represents miscalibration.

To estimate the expected accuracy from finite samples, we group predictions into M interval bins (each of size $1/M$) and calculate the accuracy of each bin. Let B_m be the set of indices of samples whose prediction confidence falls into the interval $I_m = (\frac{m-1}{M}, \frac{m}{M}]$. The accuracy of B_m is

$$\text{acc}(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} \mathbf{1}(\hat{y}_i = y_i),$$

where $\hat{y}_i$ and y_i are the predicted and true class labels for sample i. Basic probability tells us that $\text{acc}(B_m)$ is an unbiased and consistent estimator of $\mathbb{P}(\hat{Y} = Y \mid \hat{P} \in I_m)$. We define the average confidence within bin B_m as

$$\text{conf}(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} \hat{p}_i,$$

where $\hat{p}_i$ is the confidence for sample i. $\text{acc}(B_m)$ and $\text{conf}(B_m)$ approximate the left-hand and right-hand sides of (1) respectively for bin B_m. Therefore, a perfectly calibrated model will have $\text{acc}(B_m) = \text{conf}(B_m)$ for all $m \in \{1, \dots, M\}$. Note that reliability diagrams do not display the proportion of samples in a given bin, and thus cannot be used to estimate how many samples are calibrated.

Expected Calibration Error (ECE). While reliability diagrams are useful visual tools, it is more convenient to have a scalar summary statistic of calibration. Since statistics comparing two distributions cannot be comprehensive, previous works have proposed variants, each with a unique emphasis. One notion of miscalibration is the difference in expectation between confidence and accuracy, i.e.

$$\mathop{\mathbb{E}}_{\hat{P}}\left[\left|\mathbb{P}\left(\hat{Y} = Y \mid \hat{P} = p\right) - p\right|\right] \tag{2}$$

Expected Calibration Error (Naeini et al., 2015) – or ECE – approximates (2) by partitioning predictions into M equally-spaced bins (similar to the reliability diagrams) and

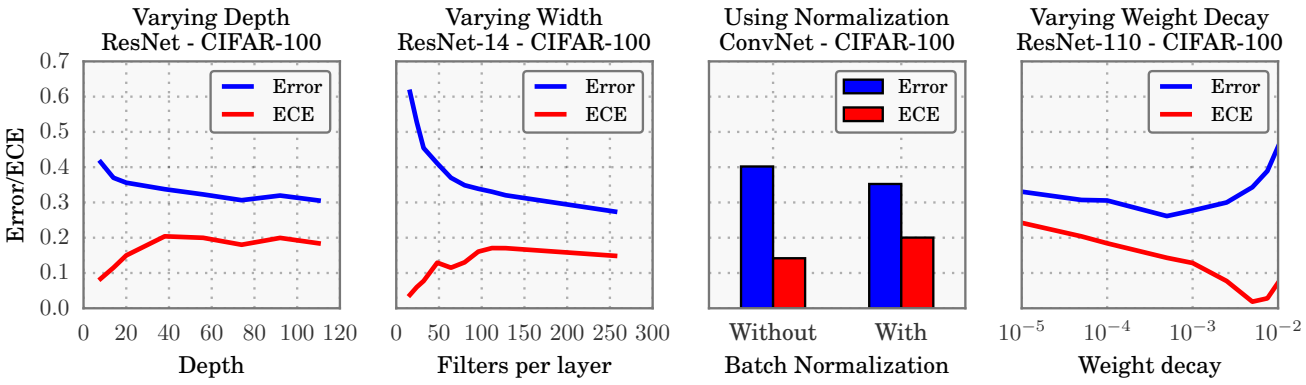

Figure 2. The effect of network depth (far left), width (middle left), Batch Normalization (middle right), and weight decay (far right) on miscalibration, as measured by ECE (lower is better).

taking a weighted average of the bins' accuracy/confidence difference. More precisely,

$$\text{ECE} = \sum_{m=1}^{M} \frac{|B_m|}{n} \Big| \text{acc}(B_m) - \text{conf}(B_m) \Big|, \qquad (3)$$

where n is the number of samples. The difference between acc and conf for a given bin represents the calibration *gap* (red bars in reliability diagrams – e.g. Figure 1). We use ECE as the primary empirical metric to measure calibration. See Section S1 for more analysis of this metric.

Maximum Calibration Error (MCE). In high-risk applications where reliable confidence measures are absolutely necessary, we may wish to minimize the worst-case deviation between confidence and accuracy:

$$\max_{p \in [0,1]} \left| \mathbb{P}\left(\hat{Y} = Y \mid \hat{P} = p\right) - p \right|. \qquad (4)$$

The Maximum Calibration Error (Naeini et al., 2015) – or MCE – estimates an upper bound of this deviation. Similarly to ECE, this approximation involves binning:

$$\text{MCE} = \max_{m \in \{1,\dots,M\}} |\text{acc}(B_m) - \text{conf}(B_m)|. \qquad (5)$$

In reliability diagrams, MCE measures the largest calibration gap (red bars) across all bins, whereas ECE measures a weighted average of all gaps. For perfectly calibrated classifiers, MCE and ECE both equal 0.

Negative log likelihood is a standard measure of a probabilistic model's quality (Friedman et al., 2001). It is also referred to as the cross entropy loss in the context of deep learning (Bengio et al., 2015). Given a probabilistic model $\hat{\pi}(Y|X)$ and n samples, NLL is defined as:

$$\mathcal{L} = -\sum_{i=1}^{n} \log(\hat{\pi}(y_i|\mathbf{x}_i)) \qquad (6)$$

It is a standard result (Friedman et al., 2001) that, in expectation, NLL is minimized if and only if $\hat{\pi}(Y|X)$ recovers the ground truth conditional distribution $\pi(Y|X)$.

3. Observing Miscalibration

The architecture and training procedures of neural networks have rapidly evolved in recent years. In this section we identify some recent changes that are responsible for the miscalibration phenomenon observed in Figure 1. Though we cannot claim causality, we find that model capacity and lack of regularization are closely related to model (mis)calibration.

Model capacity. The model capacity of neural networks has increased at a dramatic pace over the past few years. It is now common to see networks with hundreds, if not thousands of layers (He et al., 2016; Huang et al., 2016) and hundreds of convolutional filters per layer (Zagoruyko & Komodakis, 2016). Recent work shows that very deep or wide models are able to generalize better than smaller ones, while exhibiting the capacity to easily fit the training set (Zhang et al., 2017).

Although increasing depth and width may reduce classification error, we observe that these increases negatively affect model calibration. Figure 2 displays error and ECE as a function of depth and width on a ResNet trained on CIFAR-100. The far left figure varies depth for a network with 64 convolutional filters per layer, while the middle left figure fixes the depth at 14 layers and varies the number of convolutional filters per layer. Though even the smallest models in the graph exhibit some degree of miscalibration, the ECE metric grows substantially with model capacity. During training, after the model is able to correctly classify (almost) all training samples, NLL can be further minimized by increasing the confidence of predictions. Increased model capacity will lower training NLL, and thus the model will be more (over)confident on average.

Batch Normalization (Ioffe & Szegedy, 2015) improves the optimization of neural networks by minimizing distribution shifts in activations within the neural network's hid-

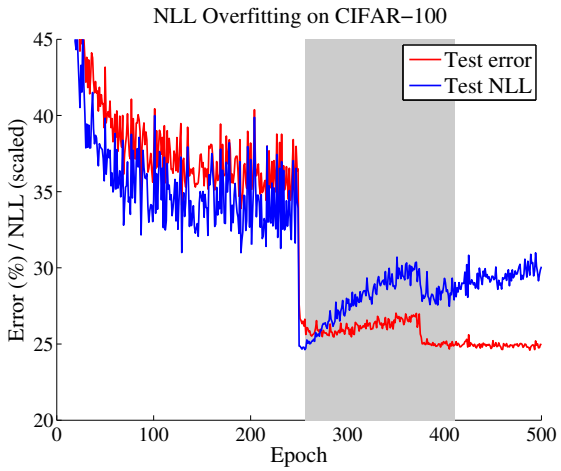

Figure 3. Test error and NLL of a 110-layer ResNet with stochastic depth on CIFAR-100 during training. NLL is scaled by a constant to fit in the figure. Learning rate drops by 10x at epochs 250 and 375. The shaded area marks between epochs at which the best validation *loss* and best validation *error* are produced.

den layers. Recent research suggests that these normalization techniques have enabled the development of very deep architectures, such as ResNets (He et al., 2016) and DenseNets (Huang et al., 2017). It has been shown that Batch Normalization improves training time, reduces the need for additional regularization, and can in some cases improve the accuracy of networks.

While it is difficult to pinpoint exactly how Batch Normalization affects the final predictions of a model, we do observe that models trained with Batch Normalization tend to be more miscalibrated. In the middle right plot of Figure 2, we see that a 6-layer ConvNet obtains worse calibration when Batch Normalization is applied, even though classification accuracy improves slightly. We find that this result holds regardless of the hyperparameters used on the Batch Normalization model (i.e. low or high learning rate, etc.).

Weight decay, which used to be the predominant regularization mechanism for neural networks, is decreasingly utilized when training modern neural networks. Learning theory suggests that regularization is necessary to prevent overfitting, especially as model capacity increases (Vapnik, 1998). However, due to the apparent regularization effects of Batch Normalization, recent research seems to suggest that models with less L2 regularization tend to generalize better (Ioffe & Szegedy, 2015). As a result, it is now common to train models with little weight decay, if any at all. The top performing ImageNet models of 2015 all use an order of magnitude less weight decay than models of previous years (He et al., 2016; Simonyan & Zisserman, 2015).

We find that training with less weight decay has a negative impact on calibration. The far right plot in Figure 2 displays training error and ECE for a 110-layer ResNet with varying amounts of weight decay. The only other forms of regularization are data augmentation and Batch Normalization. We observe that calibration and accuracy are not optimized by the same parameter setting. While the model exhibits both over-regularization and under-regularization with respect to classification error, it does not appear that calibration is negatively impacted by having too much weight decay. Model calibration continues to improve when more regularization is added, well after the point of achieving optimal accuracy. The slight uptick at the end of the graph may be an artifact of using a weight decay factor that impedes optimization.

NLL can be used to indirectly measure model calibration. In practice, we observe *a disconnect between NLL and accuracy*, which may explain the miscalibration in Figure 2. This disconnect occurs because neural networks can *overfit to NLL without overfitting to the 0/1 loss*. We observe this trend in the training curves of some miscalibrated models. Figure 3 shows test error and NLL (rescaled to match error) on CIFAR-100 as training progresses. Both error and NLL immediately drop at epoch 250, when the learning rate is dropped; however, NLL overfits during the remainder of training. Surprisingly, overfitting to NLL is beneficial to classification accuracy. On CIFAR-100, test error drops from 29% to 27% in the region where NLL overfits. This phenomenon renders a concrete explanation of miscalibration: the network learns better classification accuracy at the expense of well-modeled probabilities.

We can connect this finding to recent work examining the generalization of large neural networks. Zhang et al. (2017) observe that deep neural networks seemingly violate the common understanding of learning theory that large models with little regularization will not generalize well. The observed disconnect between NLL and 0/1 loss suggests that these high capacity models are not necessarily immune from overfitting, but rather, overfitting manifests in probabilistic error rather than classification error.

4. Calibration Methods

In this section, we first review existing calibration methods, and introduce new variants of our own. All methods are post-processing steps that produce (calibrated) probabilities. Each method requires a hold-out validation set, which in practice can be the same set used for hyperparameter tuning. We assume that the training, validation, and test sets are drawn from the same distribution.

4.1. Calibrating Binary Models

We first introduce calibration in the binary setting, i.e. $\mathcal{Y} = \{0, 1\}$. For simplicity, throughout this subsection,

we assume the model outputs only the confidence for the positive class.[1] Given a sample $\mathbf{x}_i$, we have access to $\hat{p}_i$ – the network's predicted probability of $y_i = 1$, as well as $z_i \in \mathbb{R}$ – which is the network's non-probabilistic output, or *logit*. The predicted probability $\hat{p}_i$ is derived from z_i using a sigmoid function σ; i.e. $\hat{p}_i = \sigma(z_i)$. Our goal is to produce a calibrated probability $\hat{q}_i$ based on y_i, $\hat{p}_i$, and z_i.

Histogram binning (Zadrozny & Elkan, 2001) is a simple non-parametric calibration method. In a nutshell, all uncalibrated predictions $\hat{p}_i$ are divided into mutually exclusive bins $B_1, \ldots, B_M$. Each bin is assigned a calibrated score θ_m; i.e. if $\hat{p}_i$ is assigned to bin B_m, then $\hat{q}_i = \theta_m$. At test time, if prediction $\hat{p}_{te}$ falls into bin B_m, then the calibrated prediction $\hat{q}_{te}$ is θ_m. More precisely, for a suitably chosen M (usually small), we first define bin boundaries $0 = a_1 \leq a_2 \leq \ldots \leq a_{M+1} = 1$, where the bin B_m is defined by the interval $(a_m, a_{m+1}]$. Typically the bin boundaries are either chosen to be equal length intervals or to equalize the number of samples in each bin. The predictions θ_i are chosen to minimize the bin-wise squared loss:

$$\min_{\theta_1,\ldots,\theta_M} \sum_{m=1}^{M} \sum_{i=1}^{n} \mathbf{1}(a_m \leq \hat{p}_i < a_{m+1}) \left(\theta_m - y_i\right)^2, \quad (7)$$

where $\mathbf{1}$ is the indicator function. Given fixed bins boundaries, the solution to (7) results in θ_m that correspond to the average number of positive-class samples in bin B_m.

Isotonic regression (Zadrozny & Elkan, 2002), arguably the most common non-parametric calibration method, learns a piecewise constant function f to transform uncalibrated outputs; i.e. $\hat{q}_i = f(\hat{p}_i)$. Specifically, isotonic regression produces f to minimize the square loss $\sum_{i=1}^{n}(f(\hat{p}_i) - y_i)^2$. Because f is constrained to be piecewise constant, we can write the optimization problem as:

$$\min_{\substack{M \\ \theta_1,\ldots,\theta_M \\ a_1,\ldots,a_{M+1}}} \sum_{m=1}^{M} \sum_{i=1}^{n} \mathbf{1}(a_m \leq \hat{p}_i < a_{m+1}) \left(\theta_m - y_i\right)^2$$

$$\text{subject to} \quad 0 = a_1 \leq a_2 \leq \ldots \leq a_{M+1} = 1,$$
$$\theta_1 \leq \theta_2 \leq \ldots \leq \theta_M.$$

where M is the number of intervals; $a_1, \ldots, a_{M+1}$ are the interval boundaries; and $\theta_1, \ldots, \theta_M$ are the function values. Under this parameterization, isotonic regression is a strict generalization of histogram binning in which the bin boundaries and bin predictions are jointly optimized.

Bayesian Binning into Quantiles (BBQ) (Naeini et al., 2015) is a extension of histogram binning using Bayesian model averaging. Essentially, BBQ marginalizes out all possible *binning schemes* to produce $\hat{q}_i$. More formally, a binning scheme s is a pair $(M, \mathcal{I})$ where M is the number of bins, and $\mathcal{I}$ is a corresponding partitioning of $[0, 1]$ into disjoint intervals $(0 = a_1 \leq a_2 \leq \ldots \leq a_{M+1} = 1)$. The parameters of a binning scheme are $\theta_1, \ldots, \theta_M$. Under this framework, histogram binning and isotonic regression both produce a single binning scheme, whereas BBQ considers a space $\mathcal{S}$ of all possible binning schemes for the validation dataset D. BBQ performs Bayesian averaging of the probabilities produced by each scheme:[2]

$$\begin{aligned}\mathbb{P}(\hat{q}_{te} \mid \hat{p}_{te}, D) &= \sum_{s \in \mathcal{S}} \mathbb{P}(\hat{q}_{te}, S = s \mid \hat{p}_{te}, D) \\ &= \sum_{s \in \mathcal{S}} \mathbb{P}(\hat{q}_{te} \mid \hat{p}_{te}, S = s, D)\, \mathbb{P}(S = s \mid D).\end{aligned}$$

where $\mathbb{P}(\hat{q}_{te} \mid \hat{p}_{te}, S = s, D)$ is the calibrated probability using binning scheme s. Using a uniform prior, the weight $\mathbb{P}(S = s \mid D)$ can be derived using Bayes' rule:

$$\mathbb{P}(S = s \mid D) = \frac{\mathbb{P}(D \mid S = s)}{\sum_{s' \in \mathcal{S}} \mathbb{P}(D \mid S = s')}.$$

The parameters $\theta_1, \ldots, \theta_M$ can be viewed as parameters of M independent binomial distributions. Hence, by placing a Beta prior on $\theta_1, \ldots, \theta_M$, we can obtain a closed form expression for the marginal likelihood $\mathbb{P}(D \mid S = s)$. This allows us to compute $\mathbb{P}(\hat{q}_{te} \mid \hat{p}_{te}, D)$ for any test input.

Platt scaling (Platt et al., 1999) is a parametric approach to calibration, unlike the other approaches. The non-probabilistic predictions of a classifier are used as features for a logistic regression model, which is trained on the validation set to return probabilities. More specifically, in the context of neural networks (Niculescu-Mizil & Caruana, 2005), Platt scaling learns scalar parameters $a, b \in \mathbb{R}$ and outputs $\hat{q}_i = \sigma(a z_i + b)$ as the calibrated probability. Parameters a and b can be optimized using the NLL loss over the validation set. It is important to note that the neural network's parameters are fixed during this stage.

4.2. Extension to Multiclass Models

For classification problems involving $K > 2$ classes, we return to the original problem formulation. The network outputs a class prediction $\hat{y}_i$ and confidence score $\hat{p}_i$ for each input $\mathbf{x}_i$. In this case, the network logits $\mathbf{z}_i$ are vectors, where $\hat{y}_i = \operatorname{argmax}_k z_i^{(k)}$, and $\hat{p}_i$ is typically derived using the softmax function σ_{SM}:

$$\sigma_{\text{SM}}(\mathbf{z}_i)^{(k)} = \frac{\exp(z_i^{(k)})}{\sum_{j=1}^{K} \exp(z_i^{(j)})}, \quad \hat{p}_i = \max_k \sigma_{\text{SM}}(\mathbf{z}_i)^{(k)}.$$

The goal is to produce a calibrated confidence $\hat{q}_i$ and (possibly new) class prediction $\hat{y}_i'$ based on y_i, $\hat{y}_i$, $\hat{p}_i$, and $\mathbf{z}_i$.

[1] This is in contrast with the setting in Section 2, in which the model produces both a class prediction and confidence.

[2] Because the validation dataset is finite, $\mathcal{S}$ is as well.

Dataset	Model	Uncalibrated	Hist. Binning	Isotonic	BBQ	Temp. Scaling	Vector Scaling	Matrix Scaling
Birds	ResNet 50	9.19%	4.34%	5.22%	4.12%	**1.85%**	3.0%	21.13%
Cars	ResNet 50	4.3%	**1.74%**	4.29%	1.84%	2.35%	2.37%	10.5%
CIFAR-10	ResNet 110	4.6%	0.58%	0.81%	**0.54%**	0.83%	0.88%	1.0%
CIFAR-10	ResNet 110 (SD)	4.12%	0.67%	1.11%	0.9%	**0.6%**	0.64%	0.72%
CIFAR-10	Wide ResNet 32	4.52%	0.72%	1.08%	0.74%	**0.54%**	0.6%	0.72%
CIFAR-10	DenseNet 40	3.28%	0.44%	0.61%	0.81%	**0.33%**	0.41%	0.41%
CIFAR-10	LeNet 5	3.02%	1.56%	1.85%	1.59%	**0.93%**	1.15%	1.16%
CIFAR-100	ResNet 110	16.53%	2.66%	4.99%	5.46%	**1.26%**	1.32%	25.49%
CIFAR-100	ResNet 110 (SD)	12.67%	2.46%	4.16%	3.58%	0.96%	**0.9%**	20.09%
CIFAR-100	Wide ResNet 32	15.0%	3.01%	5.85%	5.77%	**2.32%**	2.57%	24.44%
CIFAR-100	DenseNet 40	10.37%	2.68%	4.51%	3.59%	1.18%	**1.09%**	21.87%
CIFAR-100	LeNet 5	4.85%	6.48%	2.35%	3.77%	**2.02%**	2.09%	13.24%
ImageNet	DenseNet 161	6.28%	4.52%	5.18%	3.51%	**1.99%**	2.24%	-
ImageNet	ResNet 152	5.48%	4.36%	4.77%	3.56%	**1.86%**	2.23%	-
SVHN	ResNet 152 (SD)	0.44%	**0.14%**	0.28%	0.22%	0.17%	0.27%	0.17%
20 News	DAN 3	8.02%	**3.6%**	5.52%	4.98%	4.11%	4.61%	9.1%
Reuters	DAN 3	0.85%	1.75%	1.15%	0.97%	0.91%	**0.66%**	1.58%
SST Binary	TreeLSTM	6.63%	1.93%	**1.65%**	2.27%	1.84%	1.84%	1.84%
SST Fine Grained	TreeLSTM	6.71%	2.09%	**1.65%**	2.61%	2.56%	2.98%	2.39%

Table 1. ECE (%) (with $M = 15$ bins) on standard vision and NLP datasets before calibration and with various calibration methods. The number following a model's name denotes the network depth.

Extension of binning methods. One common way of extending binary calibration methods to the multiclass setting is by treating the problem as K one-versus-all problems (Zadrozny & Elkan, 2002). For $k = 1, \ldots, K$, we form a binary calibration problem where the label is $\mathbf{1}(y_i = k)$ and the predicted probability is $\sigma_{\text{SM}}(\mathbf{z}_i)^{(k)}$. This gives us K calibration models, each for a particular class. At test time, we obtain an unnormalized probability vector $[\hat{q}_i^{(1)}, \ldots, \hat{q}_i^{(K)}]$, where $\hat{q}_i^{(k)}$ is the calibrated probability for class k. The new class prediction $\hat{y}_i'$ is the argmax of the vector, and the new confidence $\hat{q}_i'$ is the max of the vector normalized by $\sum_{k=1}^{K} \hat{q}_i^{(k)}$. This extension can be applied to histogram binning, isotonic regression, and BBQ.

Matrix and vector scaling are two multi-class extensions of Platt scaling. Let $\mathbf{z}_i$ be the *logits vector* produced before the softmax layer for input $\mathbf{x}_i$. *Matrix scaling applies* a linear transformation $\mathbf{W}\mathbf{z}_i + \mathbf{b}$ to the logits:

$$\begin{aligned} \hat{q}_i &= \max_k \sigma_{\text{SM}}(\mathbf{W}\mathbf{z}_i + \mathbf{b})^{(k)}, \\ \hat{y}_i' &= \operatorname*{argmax}_k (\mathbf{W}\mathbf{z}_i + \mathbf{b})^{(k)}. \end{aligned} \tag{8}$$

The parameters $\mathbf{W}$ and $\mathbf{b}$ are optimized with respect to NLL on the validation set. As the number of parameters for matrix scaling grows quadratically with the number of classes K, we define *vector scaling* as a variant where $\mathbf{W}$ is restricted to be a diagonal matrix.

Temperature scaling, the simplest extension of Platt scaling, uses a single scalar parameter $T > 0$ for all classes. Given the logit vector $\mathbf{z}_i$, the new confidence prediction is

$$\hat{q}_i = \max_k \sigma_{\text{SM}}(\mathbf{z}_i / T)^{(k)}. \tag{9}$$

T is called the temperature, and it "softens" the softmax (i.e. raises the output entropy) with $T > 1$. As $T \to \infty$, the probability $\hat{q}_i$ approaches $1/K$, which represents maximum uncertainty. With $T = 1$, we recover the original probability $\hat{p}_i$. As $T \to 0$, the probability collapses to a point mass (i.e. $\hat{q}_i = 1$). T is optimized with respect to NLL on the validation set. Because the parameter T does not change the maximum of the softmax function, the class prediction $\hat{y}_i'$ remains unchanged. In other words, *temperature scaling does not affect the model's accuracy.*

Temperature scaling is commonly used in settings such as knowledge distillation (Hinton et al., 2015) and statistical mechanics (Jaynes, 1957). To the best of our knowledge, we are not aware of any prior use in the context of calibrating probabilistic models.[3] The model is equivalent to maximizing the entropy of the output probability distribution subject to certain constraints on the logits (see Section S2).

4.3. Other Related Works

Calibration and confidence scores have been studied in various contexts in recent years. Kuleshov & Ermon (2016) study the problem of calibration in the online setting, where the inputs can come from a potentially adversarial source. Kuleshov & Liang (2015) investigate how to produce calibrated probabilities when the output space is a structured object. Lakshminarayanan et al. (2016) use ensembles of networks to obtain uncertainty estimates. Pereyra et al. (2017) penalize overconfident predictions as a form of regularization. Hendrycks & Gimpel (2017) use confidence

[3] To highlight the connection with prior works we define temperature scaling in terms of $\frac{1}{T}$ instead of a multiplicative scalar.

scores to determine if samples are out-of-distribution.

Bayesian neural networks (Denker & Lecun, 1990; MacKay, 1992) return a probability distribution over outputs as an alternative way to represent model uncertainty. Gal & Ghahramani (2016) draw a connection between Dropout (Srivastava et al., 2014) and model uncertainty, claiming that sampling models with dropped nodes is a way to estimate the probability distribution over all possible models for a given sample. Kendall & Gal (2017) combine this approach with a model that outputs a predictive mean and variance for each data point. This notion of uncertainty is not restricted to classification problems. In contrast, our framework, which does not require model sampling, returns a confidence for a given output rather than returning a distribution of possible outputs.

5. Results

We apply the calibration methods in Section 4 to image classification and document classification neural networks. For image classification we use 6 datasets:

1. Caltech-UCSD Birds (Welinder et al., 2010): 200 bird species. 5994/2897/2897 images for train/validation/test sets.
2. Stanford Cars (Krause et al., 2013): 196 classes of cars by make, model, and year. 8041/4020/4020 images for train/validation/test.
3. ImageNet 2012 (Deng et al., 2009): Natural scene images from 1000 classes. 1.3 million/25,000/25,000 images for train/validation/test.
4. CIFAR-10/CIFAR-100 (Krizhevsky & Hinton, 2009): Color images (32×32) from 10/100 classes. 45,000/5,000/10,000 images for train/validation/test.
5. Street View House Numbers (SVHN) (Netzer et al., 2011): 32×32 colored images of cropped out house numbers from Google Street View. 604,388/6,000/26,032 images for train/validation/test.

We train state-of-the-art convolutional networks: ResNets (He et al., 2016), ResNets with stochastic depth (SD) (Huang et al., 2016), Wide ResNets (Zagoruyko & Komodakis, 2016), and DenseNets (Huang et al., 2017). We use the data preprocessing, training procedures, and hyperparameters as described in each paper. For Birds and Cars, we fine-tune networks pretrained on ImageNet.

For document classification we experiment with 4 datasets:

1. 20 News: News articles, partitioned into 20 categories by content. 9034/2259/7528 documents for train/validation/test.
2. Reuters: News articles, partitioned into 8 categories by topic. 4388/1097/2189 documents for train/validation/test.
3. Stanford Sentiment Treebank (SST) (Socher et al., 2013): Movie reviews, represented as sentence parse trees that are annotated by sentiment. Each sample includes a coarse binary label and a fine grained 5-class label. As described in (Tai et al., 2015), the training/validation/test sets contain 6920/872/1821 documents for binary, and 544/1101/2210 for fine-grained.

On 20 News and Reuters, we train Deep Averaging Networks (DANs) (Iyyer et al., 2015) with 3 feed-forward layers and Batch Normalization. These networks obtain competitive accuracy using the optimization hyperparameters suggested by the original paper. On SST, we train TreeLSTMs (Long Short Term Memory) (Tai et al., 2015) using the default settings in the authors' code.

Calibration Results. Table 1 displays model calibration, as measured by ECE (with $M = 15$ bins), before and after applying the various methods (see Section S3 for MCE, NLL, and error tables). It is worth noting that most datasets and models experience some degree of miscalibration, with ECE typically between 4 to 10%. This is not architecture specific: we observe miscalibration on convolutional networks (with and without skip connections), recurrent networks, and deep averaging networks. The two notable exceptions are SVHN and Reuters, both of which experience ECE values below 1%. Both of these datasets have very low error (1.98% and 2.97%, respectively); and therefore the ratio of ECE to error is comparable to other datasets.

Our most important discovery is the *surprising effectiveness of temperature scaling* despite its remarkable simplicity. Temperature scaling outperforms all other methods on the vision tasks, and performs comparably to other methods on the NLP datasets. What is perhaps even more surprising is that temperature scaling outperforms the vector and matrix Platt scaling variants, which are strictly more general methods. In fact, vector scaling recovers essentially the same solution as temperature scaling – the learned vector has nearly constant entries, and therefore is no different than a scalar transformation. In other words, network miscalibration is intrinsically low dimensional.

The only dataset that temperature scaling does not calibrate is the Reuters dataset. In this instance, only one of the above methods is able to improve calibration. Because this dataset is well-calibrated to begin with (ECE $\leq 1\%$), there is not much room for improvement with any method, and post-processing may not even be necessary to begin with. It is also possible that our measurements are affected by dataset split or by the particular binning scheme.

Matrix scaling performs poorly on datasets with hundreds of classes (i.e. Birds, Cars, and CIFAR-100), and fails to converge on the 1000-class ImageNet dataset. This is expected, since the number of parameters scales quadrat-

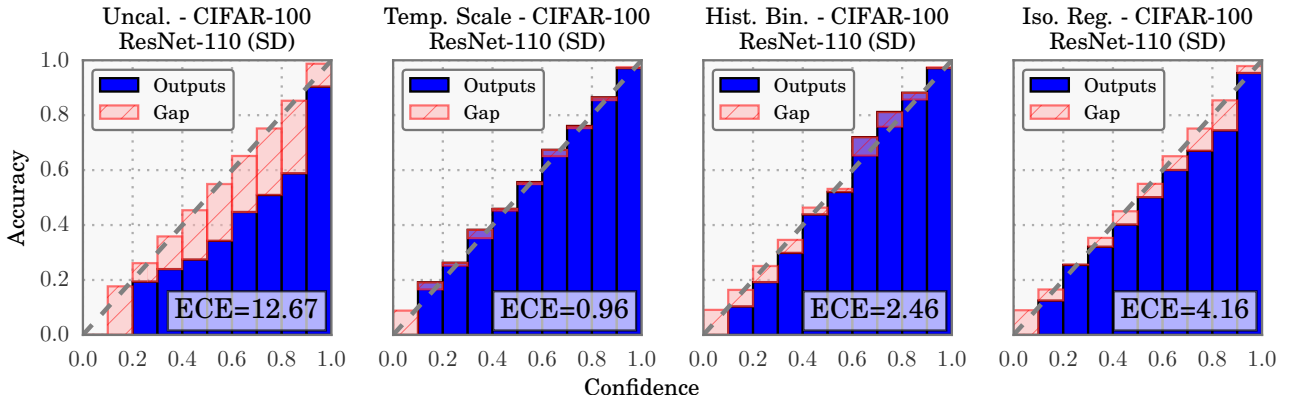

Figure 4. Reliability diagrams for CIFAR-100 before (far left) and after calibration (middle left, middle right, far right).

ically with the number of classes. Any calibration model with tens of thousands (or more) parameters will overfit to a small validation set, even when applying regularization.

Binning methods improve calibration on most datasets, but do not outperform temperature scaling. Additionally, binning methods tend to change class predictions which hurts accuracy (see Section S3). Histogram binning, the simplest binning method, typically outperforms isotonic regression and BBQ, despite the fact that both methods are strictly more general. This further supports our finding that calibration is best corrected by simple models.

Reliability diagrams. Figure 4 contains reliability diagrams for 110-layer ResNets on CIFAR-100 before and after calibration. From the far left diagram, we see that the uncalibrated ResNet tends to be overconfident in its predictions. We then can observe the effects of temperature scaling (middle left), histogram binning (middle right), and isotonic regression (far right) on calibration. All three displayed methods produce much better confidence estimates. Of the three methods, temperature scaling most closely recovers the desired diagonal function. Each of the bins are well calibrated, which is remarkable given that all the probabilities were modified by only a single parameter. We include reliability diagrams for other datasets in Section S4.

Computation time. All methods scale linearly with the number of validation set samples. Temperature scaling is by far the fastest method, as it amounts to a one-dimensional convex optimization problem. Using a conjugate gradient solver, the optimal temperature can be found in 10 iterations, or a fraction of a second on most modern hardware. In fact, even a naive line-search for the optimal temperature is faster than any of the other methods. The computational complexity of vector and matrix scaling are linear and quadratic respectively in the number of classes, reflecting the number of parameters in each method. For CIFAR-100 ($K = 100$), finding a near-optimal vector scaling solution with conjugate gradient descent requires at least 2 orders of magnitude more time. Histogram binning and isotonic regression take an order of magnitude longer than temperature scaling, and BBQ takes roughly 3 orders of magnitude more time.

Ease of implementation. BBQ is arguably the most difficult to implement, as it requires implementing a model averaging scheme. While all other methods are relatively easy to implement, temperature scaling may arguably be the most straightforward to incorporate into a neural network pipeline. In Torch7 (Collobert et al., 2011), for example, we implement temperature scaling by inserting a `nn.MulConstant` between the logits and the softmax, whose parameter is $1/T$. We set $T = 1$ during training, and subsequently find its optimal value on the validation set.

6. Conclusion

Modern neural networks exhibit a strange phenomenon: probabilistic error and miscalibration worsen even as classification error is reduced. We have demonstrated that recent advances in neural network architecture and training – model capacity, normalization, and regularization – have strong effects on network calibration. It remains future work to understand why these trends affect calibration while improving accuracy. Nevertheless, simple techniques can effectively remedy the miscalibration phenomenon in neural networks. Temperature scaling is the simplest, fastest, and most straightforward of the methods, and surprisingly is often the most effective.

Acknowledgments

The authors are supported in part by the III-1618134, III-1526012, and IIS-1149882 grants from the National Science Foundation, as well as the Bill and Melinda Gates Foundation and the Office of Naval Research.

References

Bengio, Yoshua, Goodfellow, Ian J, and Courville, Aaron. Deep learning. *Nature*, 521:436–444, 2015.

Bojarski, Mariusz, Del Testa, Davide, Dworakowski, Daniel, Firner, Bernhard, Flepp, Beat, Goyal, Prasoon, Jackel, Lawrence D, Monfort, Mathew, Muller, Urs, Zhang, Jiakai, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

Caruana, Rich, Lou, Yin, Gehrke, Johannes, Koch, Paul, Sturm, Marc, and Elhadad, Noemie. Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. In *KDD*, 2015.

Collobert, Ronan, Kavukcuoglu, Koray, and Farabet, Clément. Torch7: A matlab-like environment for machine learning. In *BigLearn Workshop, NIPS*, 2011.

Cosmides, Leda and Tooby, John. Are humans good intuitive statisticians after all? rethinking some conclusions from the literature on judgment under uncertainty. *cognition*, 58(1):1–73, 1996.

DeGroot, Morris H and Fienberg, Stephen E. The comparison and evaluation of forecasters. *The statistician*, pp. 12–22, 1983.

Deng, Jia, Dong, Wei, Socher, Richard, Li, Li-Jia, Li, Kai, and Fei-Fei, Li. Imagenet: A large-scale hierarchical image database. In *CVPR*, pp. 248–255, 2009.

Denker, John S and Lecun, Yann. Transforming neural-net output levels to probability distributions. In *NIPS*, pp. 853–859, 1990.

Friedman, Jerome, Hastie, Trevor, and Tibshirani, Robert. *The elements of statistical learning*, volume 1. Springer series in statistics Springer, Berlin, 2001.

Gal, Yarin and Ghahramani, Zoubin. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *ICML*, 2016.

Girshick, Ross. Fast r-cnn. In *ICCV*, pp. 1440–1448, 2015.

Hannun, Awni, Case, Carl, Casper, Jared, Catanzaro, Bryan, Diamos, Greg, Elsen, Erich, Prenger, Ryan, Satheesh, Sanjeev, Sengupta, Shubho, Coates, Adam, et al. Deep speech: Scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412.5567*, 2014.

He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. Deep residual learning for image recognition. In *CVPR*, pp. 770–778, 2016.

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

Hinton, Geoffrey, Vinyals, Oriol, and Dean, Jeff. Distilling the knowledge in a neural network. 2015.

Huang, Gao, Sun, Yu, Liu, Zhuang, Sedra, Daniel, and Weinberger, Kilian. Deep networks with stochastic depth. In *ECCV*, 2016.

Huang, Gao, Liu, Zhuang, Weinberger, Kilian Q, and van der Maaten, Laurens. Densely connected convolutional networks. In *CVPR*, 2017.

Ioffe, Sergey and Szegedy, Christian. Batch normalization: Accelerating deep network training by reducing internal covariate shift. 2015.

Iyyer, Mohit, Manjunatha, Varun, Boyd-Graber, Jordan, and Daumé III, Hal. Deep unordered composition rivals syntactic methods for text classification. In *ACL*, 2015.

Jaynes, Edwin T. Information theory and statistical mechanics. *Physical review*, 106(4):620, 1957.

Jiang, Xiaoqian, Osl, Melanie, Kim, Jihoon, and Ohno-Machado, Lucila. Calibrating predictive model estimates to support personalized medicine. *Journal of the American Medical Informatics Association*, 19(2):263–274, 2012.

Kendall, Alex and Cipolla, Roberto. Modelling uncertainty in deep learning for camera relocalization. 2016.

Kendall, Alex and Gal, Yarin. What uncertainties do we need in bayesian deep learning for computer vision? *arXiv preprint arXiv:1703.04977*, 2017.

Krause, Jonathan, Stark, Michael, Deng, Jia, and Fei-Fei, Li. 3d object representations for fine-grained categorization. In *IEEE Workshop on 3D Representation and Recognition (3dRR)*, Sydney, Australia, 2013.

Krizhevsky, Alex and Hinton, Geoffrey. Learning multiple layers of features from tiny images, 2009.

Kuleshov, Volodymyr and Ermon, Stefano. Reliable confidence estimation via online learning. *arXiv preprint arXiv:1607.03594*, 2016.

Kuleshov, Volodymyr and Liang, Percy. Calibrated structured prediction. In *NIPS*, pp. 3474–3482, 2015.

Lakshminarayanan, Balaji, Pritzel, Alexander, and Blundell, Charles. Simple and scalable predictive uncertainty estimation using deep ensembles. *arXiv preprint arXiv:1612.01474*, 2016.

LeCun, Yann, Bottou, Léon, Bengio, Yoshua, and Haffner, Patrick. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

MacKay, David JC. A practical bayesian framework for backpropagation networks. *Neural computation*, 4(3): 448–472, 1992.

Naeini, Mahdi Pakdaman, Cooper, Gregory F, and Hauskrecht, Milos. Obtaining well calibrated probabilities using bayesian binning. In *AAAI*, pp. 2901, 2015.

Netzer, Yuval, Wang, Tao, Coates, Adam, Bissacco, Alessandro, Wu, Bo, and Ng, Andrew Y. Reading digits in natural images with unsupervised feature learning. In *Deep Learning and Unsupervised Feature Learning Workshop, NIPS*, 2011.

Niculescu-Mizil, Alexandru and Caruana, Rich. Predicting good probabilities with supervised learning. In *ICML*, pp. 625–632, 2005.

Pereyra, Gabriel, Tucker, George, Chorowski, Jan, Kaiser, Łukasz, and Hinton, Geoffrey. Regularizing neural networks by penalizing confident output distributions. *arXiv preprint arXiv:1701.06548*, 2017.

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

Simonyan, Karen and Zisserman, Andrew. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

Socher, Richard, Perelygin, Alex, Wu, Jean, Chuang, Jason, Manning, Christopher D., Ng, Andrew, and Potts, Christopher. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*, pp. 1631–1642, 2013.

Srivastava, Nitish, Hinton, Geoffrey, Krizhevsky, Alex, Sutskever, Ilya, and Salakhutdinov, Ruslan. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

Srivastava, Rupesh Kumar, Greff, Klaus, and Schmidhuber, Jürgen. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015.

Tai, Kai Sheng, Socher, Richard, and Manning, Christopher D. Improved semantic representations from tree-structured long short-term memory networks. 2015.

Vapnik, Vladimir N. *Statistical Learning Theory*. Wiley-Interscience, 1998.

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.

Xiong, Wayne, Droppo, Jasha, Huang, Xuedong, Seide, Frank, Seltzer, Mike, Stolcke, Andreas, Yu, Dong, and Zweig, Geoffrey. Achieving human parity in conversational speech recognition. *arXiv preprint arXiv:1610.05256*, 2016.

Zadrozny, Bianca and Elkan, Charles. Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers. In *ICML*, pp. 609–616, 2001.

Zadrozny, Bianca and Elkan, Charles. Transforming classifier scores into accurate multiclass probability estimates. In *KDD*, pp. 694–699, 2002.

Zagoruyko, Sergey and Komodakis, Nikos. Wide residual networks. In *BMVC*, 2016.

Zhang, Chiyuan, Bengio, Samy, Hardt, Moritz, Recht, Benjamin, and Vinyals, Oriol. Understanding deep learning requires rethinking generalization. In *ICLR*, 2017.