Adaptive confidence calibration

An in-training parameter tuning method

Jonathan Pearce

School of Computer Science

McGill University, Montreal

August 2021

A thesis submitted to McGill University in partial fulfillment of the requirements of the degree of Master of Science in Computer Science

Acknowledgments

I would like to thank all the people who have influenced me during the course of my graduate studies. I want to thank my supervisor David Meger for his support, encouragement and always insightful comments. I would like to thank the members of the Mobile Robotics Laboratory, all of whom were always open to discuss anything and were very helpful whenever I had a problem. Finally, I would like to thank my parents, Ann and Glen, and my sister, Katherine, for all their support, throughout my studies.

Abstract

Confidence calibration is a quickly growing area of research in deep learning, including computer vision applications. New model architectures and loss functions are being introduced to improve model calibration, an important topic for safety critical AI applications. In order to evaluate these new methods, they are frequently compared to simple baselines, which serve as a comparison and help measure new methods' effectiveness. Popular baselines for evaluating model confidence calibration include label smoothing, mixup and dropout. Despite these methods frequent use and simple implementations, the parameter values that define them are rarely validated, and are usually a default value. This thesis demonstrates the danger in using these default values for calibration benchmarks on common datasets; poor model calibration, specifically, a model that is purely over- or underconfident. We present an adaptive framework that can adjust the parameter value of these baseline methods during training based on validation accuracy and confidence to maintain good model calibration and balance over- and underconfidence. Experiments with the CIFAR-10 and CIFAR-100 image classification datasets show that our approach improves model calibration compared to using these popular methods with default values and we achieve good model calibration regardless of model architecture and dataset. Further, our analysis provides a comparison between these baseline methods that can be considered for future confidence calibration research.

Abrégé

L'étalonnage de la confiance est un domaine de recherche en croissance rapide dans le domaine de l'apprentissage en profondeur, y compris les applications de vision par ordinateur. De nouvelles architectures de modèles et fonctions de perte sont introduites pour améliorer l'étalonnage des modèles, un sujet important pour les applications d'IA critiques pour la sécurité. Afin d'évaluer ces nouvelles méthodes, elles sont fréquemment comparées à des bases de référence simples, qui servent de comparaison et permettent de mesurer l'efficacité des nouvelles méthodes. Les lignes de base courantes pour évaluer l'étalonnage de la confiance du modèle incluent le lissage des étiquettes, la confusion et l'abandon. Malgré l'utilisation fréquente de ces méthodes et des implémentations simples, les valeurs des paramètres qui les définissent sont rarement validées, et sont généralement une valeur par défaut. Cette thèse démontre le danger d'utiliser ces valeurs par défaut pour des benchmarks de calibration sur des jeux de données communs ; un mauvais calibrage du modèle, en particulier un modèle qui est purement trop ou pas assez confiant. Nous présentons un cadre adaptatif qui peut ajuster la valeur des paramètres de ces méthodes de base au cours de la formation en fonction de la précision et de la confiance de validation pour maintenir un bon étalonnage du modèle et équilibrer la confiance excessive et insuffisante. Les expériences avec les ensembles de données de classification d'images CIFAR-10 et CIFAR-100 montrent que notre

approche améliore l'étalonnage du modèle par rapport à l'utilisation de ces méthodes populaires avec des valeurs par défaut et nous obtenons de bonnes bases de référence d'étalonnage indépendamment de l'architecture du modèle et de l'ensemble de données. De plus, notre analyse fournit une comparaison entre ces méthodes de base qui peuvent être envisagées pour les futures recherches sur l'étalonnage de la confiance.

Contents

List of Figures

List of Tables

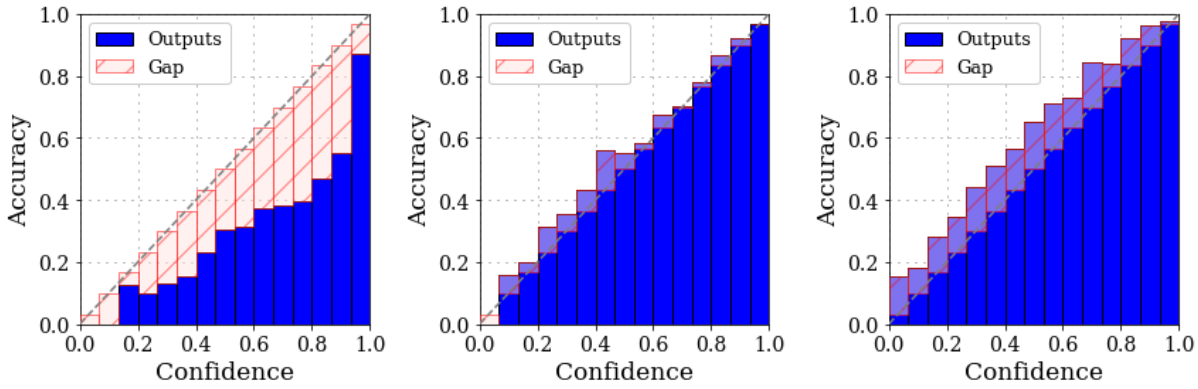
Chapter 1

Introduction

In machine learning applications that involve probabilistic predictions, knowing that the probabilities outputted by the system reflect the true chance of an event occurring is important. A machine learning model that has accurate prediction confidences is considered well calibrated and is less harmful to use in real world applications, than a model that is not well calibrated. Calibration is considered the most important property of a predictive model in certain applications [Alba et al., 2017]. The most common definition for calibration is to ask if we observe a predicted outcome R% of the time when the prediction is made with R% confidence. For example, in a clinical health setting, if an image classification model predicts that a breast cancer patient has an 80% probability of residual tumor tissue then the observed frequency of tumor tissue should be approximately 80 out of 100 patients with the same predicted probability. Typically in deep learning applications, these output probabilities are obtained with a softmax function in the last layer of the neural network. A poorly calibrated model can be very costly to the people interacting with it, as well as the people operating and maintaining the model. Confidence calibration has been identified

as one of the most significant challenges for deep learning applications [Sünderhauf et al., 2018]. Machine learning algorithms continue to replace humans in decision-making pipelines and are being deployed in high risk fields such as autonomous driving [Levinson et al., 2011] and healthcare [Miotto et al., 2016]. Despite impressive accuracies in supervised learning tasks, such as image classification, modern deep neural networks are typically poorly calibrated; their predictions' are significantly overconfident when trained with hard targets and standard data augmentation methods [Guo et al., 2017]. Models that exhibit poor calibration, specifically overconfidence can be harmful or offensive when used in practical high risk settings [Amodei et al., 2016]. Therefore it is crucial for deep learning models to be well calibrated.

Since the observations that modern deep neural networks are poorly calibrated when trained using a default approach [Guo et al., 2017] there has been an increase into calibration-based research with new model architectures designed specifically to minimize calibration error [Xing et al., 2020], loss functions that prevent significant overconfidence [Mukhoti et al., 2020] and methods that calibrate individual data classes independently [Wen et al., 2021]. Most of this new research utilizes popular baseline methods that have been shown to be well calibrated (when used with properly validated parameter values), as way to compare and evaluate these more novel methods. Like all empirically driven research, strong and reliable baselines are fundamental to developing and evaluating successful new methods. There are a number of common and popular calibration baseline methods for computer vision classification tasks: label smoothing [Szegedy et al., 2016], a label augmentation technique; mixup [Zhang et al., 2018], a data augmentation technique; and dropout [Srivastava et al., 2014], a method that helps prevents over-fitting by modifying a model's architecture during training. These three methods were originally introduced as regularization techniques which

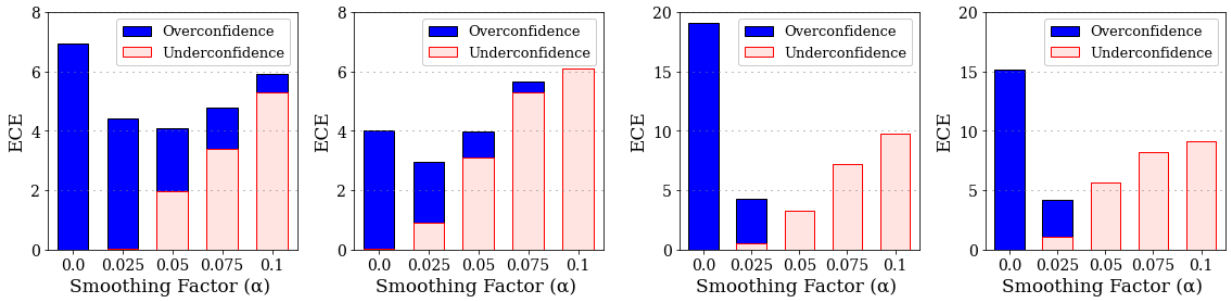


(a) SF = 0 (Hard Targets) (b) SF = 0.05 (c) SF = 0.1

Figure 1.1: Reliability diagram analysis of smoothing factor (SF) impact on model calibration for VGG11 model trained on CIFAR-100 dataset.

improved model accuracy in classification tasks. Later research, found that these methods are all able to improve calibration compared to default training methods [Müller et al., 2019, Thulasidasan et al., 2019], while maintaining the increase in model accuracy. With respect to calibration, these methods' efficacy depends on their parameter values which, despite their frequent use, are typically set to a default value without validation [Xing et al., 2020, Müller et al., 2019]. When their parameters are validated, the procedure is rarely detailed [Mukhoti et al., 2020, Thulasidasan et al., 2019, Lukasik et al., 2020]. Without a thorough validation procedure for the parameters that define these methods, that accounts for the dataset, model architecture and training procedure, baseline benchmarks for confidence calibration can vary greatly in calibration error [Carratino et al., 2020] and make it difficult to evaluate the significance of newer methods.

Of these popular benchmarking methods, label smoothing is most widely used. Label smoothing is simple to implement and it typically improves a model's accuracy, calibration and ability to generalize to new data [Müller et al., 2019]. Figure 1.1a illustrates the over-confidence issue of deep neural networks trained with a standard learning procedure (no



(a) CIFAR10 - VGG (b) CIFAR10 - ResNet (c) CIFAR100 - VGG (d) CIFAR100 - ResNet

Figure 1.2: Analysis of smoothing factor impact on model over- and underconfidence. Overconfidence = ECEOC and underconfidence = ECEOC are defined in section 4.1. Smoothing factor = 0 is equivalent to training with hard targets.

label smoothing), within each confidence bin, the average output confidence is greater than the accuracy of predictions in that bin. Figure 1.1b shows how label smoothing can reduce this overconfidence issue and produce a well calibrated model. However, Figure 1.1c shows the risk of a poorly chosen smoothing factor for label smoothing, the model's overconfidence issue is simply replaced with significant underconfidence. Balancing over- and underconfidence depends on the smoothing factor, and the optimal smoothing factor value for model calibration can depend on both model architecture and dataset as seen in Figure 1.2. Table 1.1 provides a brief summary of comparable label smoothing results, with chosen smoothing factor and validation procedure (if available). Despite its frequent use, the smoothing factor is typically set to a default value without validation. When the smoothing factor is validated, only a few values are considered. Validating the smoothing factor can be expensive, as some models can take days to train, however, an improper smoothing factor can result in poor performance. These same issues arise with mixup and dropout and the parameter values that define these methods.

In this thesis we demonstrate the potential danger of improperly validating the parameter

Table 1.1: Label smoothing - survey of literature, smoothing factor (SF) validation procedure and calibration results for experiments on CIFAR-100 dataset

Model	**Paper**	**SF Validation**	**SF Final**	**ECE(%)**
ResNet50	[Mukhoti et al., 2020]	{0.05, 0.1}	0.05	7.81
	[Xing et al., 2020]	N/A	N/A	3.3
ResNet56	[Müller et al., 2019]	N/A	0.05	2.4
	[Zhang et al., 2020]	N/A	0.1	3.35 ± 0.86
ResNet110	[Mukhoti et al., 2020]	{0.05, 0.1}	0.05	11.02
	[Zhang et al., 2020]	N/A	0.1	2.32 ± 1.03

values for popular baseline calibration methods and using default values in calibration experiments. We find that although these methods generally decrease calibration error compared to standard training procedures, using incorrect parameters can fail to alleviate the model overconfidence issue, and can in some cases create an underconfidence problem. We provide a decomposition of Expected Calibration Error (ECE) that provides metrics to measure over- and underconfidence calibration error for classification models. These measures of over- and underconfidence enable us to create an adaptive parameter framework for model calibration. This framework circumvents the issue of parameter validation in calibration methods. It works by adjusting the respective parameters of the calibration methods during training based on validation accuracy and confidence statistics, to maintain good model calibration. We extend this adaptive framework further with label smoothing, developing and testing a class adaptive parameter framework, where each class has its own smoothing factor that adjusts during training based on class specific statistics. Finally, we present an additional calibration metric, Confidence Balance (CB) that evaluates how well a model is balancing over- and underconfidence and justify that this metric should be used in addition to calibration error metrics to evaluate model calibration. We evaluate our adaptive calibration

framework and compare to the baseline methods with default parameter values. We conduct experiments on the CIFAR-10 and CIFAR-100 image classification datasets, evaluating model accuracy, calibration error and over- and underconfidence balance. We find that our adaptive calibration framework is an effective method for validating the parameter values of calibration methods during training. Our methods improve model calibration compared to baseline methods with fixed default values. We also gain further insights into the strengths and weaknesses of using label smoothing, mixup and dropout for calibration research.

1.1 Contributions of this work

The contribution of this work is the formulation of an adaptive parameter framework for calibrating deep neural networks during training. This framework can be used in calibration research to ensure that methods such as label smoothing and mixup are well calibrated and therefore provide a proper comparison and evaluation of more novel methods. This framework adjusts parameter values during training to guarantee well calibrated models, regardless of model architecture, dataset and training setup. Our framework adjusts these parameters based on model over- and underconfidence, an often overlooked aspect of calibration. We provide a decomposition of the well used metric, Expected Calibration Error (ECE) into two terms that estimate model over- and underconfidence. We present a new evaluation metric to help measure a models' balance of over- and underconfidence, this metric can be used in tandem with calibration error metrics to evaluate a models' true calibration. Finally our results and analysis provide new insights for using label smoothing, mixup and dropout for calibration research.

1.2 Thesis Outline

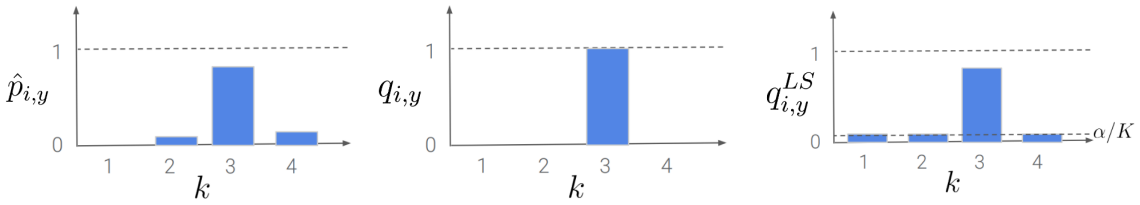
This work is organized as follows. In Chapter 2, we present background information for the methods that we use in our experimental methods; label smoothing, mixup and dropout. We also provide background on calibration evaluation methods. In Chapter 3, we present recent calibration research focusing on methods for in-training calibration, we further discuss how methods such as label smoothing, mixup and dropout have been used as benchmarks in calibration research as well as new methods that extend these basic approaches. In Chapter 4, we introduce our adaptive framework for parameter adjustments during training and show how this framework can be applied to label smoothing, mixup and last-layer dropout. Finally, in Chapter 5 we discuss the results obtained and in Chapter 6 we suggest possible directions for future work.

Chapter 2

Background

In this chapter, we present the background on the learning methods and calibration metrics used in the methods and results sections of this thesis. The first three subsections describe learning methods for training; label smoothing, mixup and dropout. Theses methods have previously been shown to produce more well calibrated models than a standard training procedure. The fourth subsection, outlines metrics for evaluating model calibration error, these metrics are typically evaluated using the validation and test sets.

Let $\mathcal{D} = \langle(\mathbf{x}_i, y_i)\rangle_{i=1}^{N}$ be a dataset consisting of N independent and identically distributed real-world images belonging to K different classes. For each sample i, $\mathbf{x}_i \in \mathcal{X}$ is the input image and $y_i \in \mathcal{Y} = \{1, 2, ..., K\}$ is the ground-truth class label. Let f_θ represent the CNN classifier f with model parameters denoted by θ. Let $\hat{p}_{i,y} = f_\theta(y|\mathbf{x}_i)$ be the confidence (computed using the softmax function) that the image $\mathbf{x}_i$ belongs to the class y. The predicted class is then $\hat{y}_i = \text{argmax}_{y \in Y} \hat{p}_{i,y}$ and the predicted class confidence is $\hat{p}_i = \max_{y \in Y} \hat{p}_{i,y}$, following the notation adopted by Mukhoti et al. [2020].



(a) model output (b) hard target (c) label smoothing target

Figure 2.1: Illustration of label smoothing

2.1 Label Smoothing

Label smoothing [Szegedy et al., 2016] improves accuracy and model calibration [Müller et al., 2019] by computing cross entropy not with the "hard" targets from the dataset, but with a weighted mixture of these targets with the uniform distribution as illustrated in Figure 2.1c. We denote the distribution of the hard target label for image $\mathbf{x}_i$ with ground truth class y_i as $\mathbf{q}_i = (q_{i,1}, ..., q_{i,K})$, where $q_{i,y} = 1$ when $y = y_i$, and $q_{i,y} = 0$ otherwise. For a model trained with hard targets we minimize the expected value of the cross-entropy loss $\mathcal{L}$ between the true targets $q_{i,y}$ the model's outputs $\hat{p}_{i,y}$

$$\mathcal{L}(\hat{p}, q) = \sum_{y=1}^{K} -q_{i,y} \log(\hat{p}_{i,y}). \tag{2.1}$$

For a network trained with label smoothing, with smoothing factor α, we instead minimize the cross-entropy between the smoothed targets $q_{i,y}^{LS}$, and the model's outputs $\hat{p}_{i,y}$. Where $q_{i,y}^{LS}$ is a weighted average of the hard targets and the uniform distribution over labels,

$$q_{i,y}^{LS} = (1 - \alpha) q_{i,y} + \alpha / K. \tag{2.2}$$

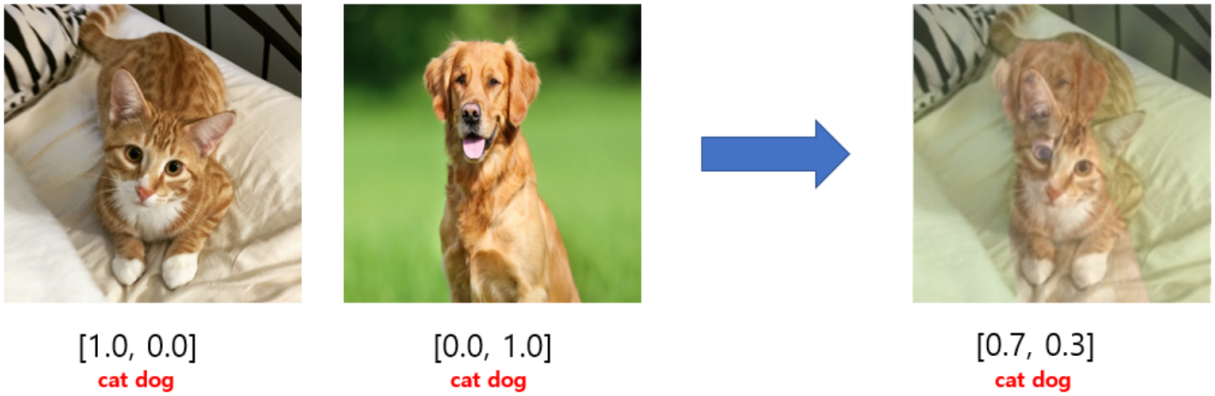


Figure 2.2: Illustration of mixup

2.2 Mixup

Mixup training [Zhang et al., 2018] is based on the principle of Vicinal Risk Minimization [Chapelle et al., 2001]: the classifier is trained not only on the training data, but also in the vicinity of each training sample. The vicinal points are generated according to the following simple rule:

$$\begin{aligned} \tilde{\mathbf{x}} &= \lambda \mathbf{x}_i + (1-\lambda)\mathbf{x}_j, \\ \tilde{\mathbf{y}} &= \lambda \mathbf{y}_i + (1-\lambda)\mathbf{y}_j, \end{aligned} \tag{2.3}$$

where $\mathbf{x}_i$ and $\mathbf{x}_j$ are two randomly sampled input points, and $\mathbf{y}_i$ and $\mathbf{y}_j$ are the label distributions of the respective images, Figure 2.2 illustrates this process. This has the effect of the empirical Dirac delta distribution,

$$P_\delta(\mathbf{x}, \mathbf{y}) = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x} = \mathbf{x}_i, \mathbf{y} = \mathbf{y}_i) \tag{2.4}$$

centered at $(\mathbf{x}_i, \mathbf{y}_i)$ being replaced with the empirical vicinal distribution

$$P_\nu(\mathbf{x}, \mathbf{y}) = \frac{1}{n} \sum_{i=1}^{n} \nu(\mathbf{x} = \mathbf{x}_i, \mathbf{y} = \mathbf{y}_i), \tag{2.5}$$

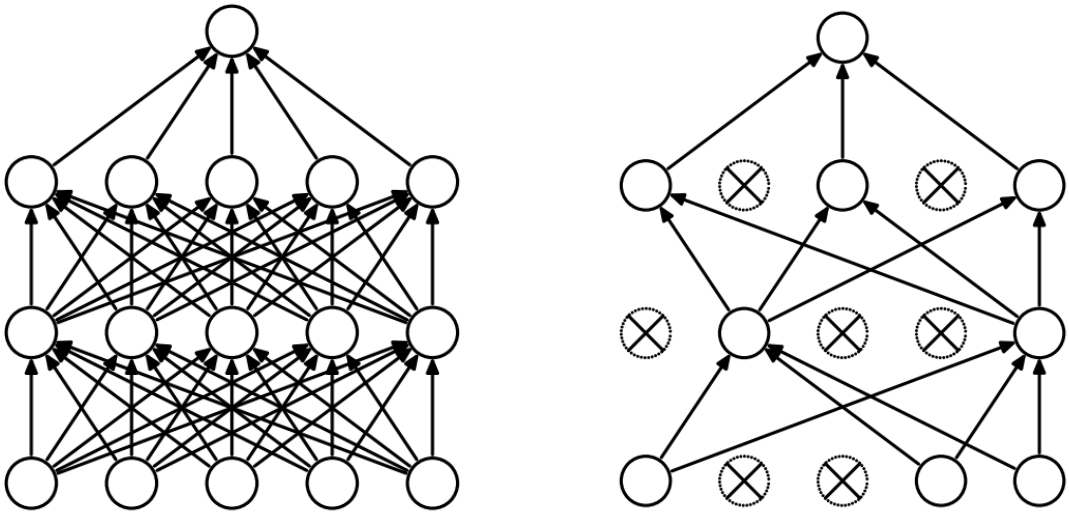
where ν is a vicinity distribution that gives the probability of finding the virtual feature-target pair $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})$ in the vicinity of the original pair $(\mathbf{x}_i, \mathbf{y}_i)$. The vicinal samples $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})$ are generated as above, and during training minimization is performed on the empirical vicinal risk using the vicinal dataset $\mathcal{D}_\nu := \{(\tilde{\mathbf{x}}_i, \tilde{\mathbf{y}}_i)\}_{i=1}^{m}$

$$R_\nu(f) = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(f(\tilde{\mathbf{x}}_i), \tilde{\mathbf{y}}_i), \tag{2.6}$$

where $\mathcal{L}$ is the standard cross-entropy loss, but calculated on the soft-labels $\tilde{\mathbf{y}}_i$ instead of hard labels. The linear interpolator $\lambda \in [0, 1]$ that determines the mixing ratio is drawn from a symmetric Beta distribution, Beta(ϵ, ϵ) at each training iteration, where ϵ is the hyperparameter that controls the strength of the interpolation between pairs of images and the associated smoothing of the training labels. $\epsilon = 0$ recovers the base case corresponding to zero-entropy training labels (hard targets, in which case the resulting image is either just $\mathbf{x}_i$ or $\mathbf{x}_j$), while a high value of ϵ ends up in always averaging the inputs and labels.

2.3 Dropout

Dropout [Srivastava et al., 2014] is used frequently in deep learning as a way to avoid over-fitting, the key idea is to randomly drop units (along with their connections) from the neural network during training. Consider a neural network with L hidden layers. Let $l \in \{1, ..., L\}$ index the hidden layers of the network. Let $\mathbf{z}^{(l)}$ denote the vector of inputs into layer l,



(a) Standard Neural Net (b) After applying dropout.

Figure 2.3: Illustration of dropout

$\mathbf{y}^{(l)}$ denote the vector of outputs from layer l ($\mathbf{y}^{(0)} = x$ is the input). $\mathbf{W}^{(l)}$ and $\mathbf{b}^{(l)}$ are the weights and biases at layer l. The feed-forward operation of a standard neural network can be described as (for $l \in \{0, ..., L-1\}$ and any hidden unit i),

$$\begin{aligned} z_i^{(l+1)} &= \mathbf{w}_i^{(l+1)} \mathbf{y}^{(l)} + b_i^{(l+1)}, \\ y_i^{(l+1)} &= f(z_i^{(l+1)}), \end{aligned} \tag{2.7}$$

where f is any activation function. With dropout, the feed-forward operation becomes

$$\begin{aligned} r_j^{(l)} &\sim \text{Bernoulli}(p) \\ \tilde{\mathbf{y}} &= \mathbf{r}^{(l)} * \mathbf{y}^{(l)} \\ z_i^{(l+1)} &= \mathbf{w}_i^{(l+1)} \tilde{\mathbf{y}}^{(l)} + b_i^{(l+1)}, \\ y_i^{(l+1)} &= f(z_i^{(l+1)}). \end{aligned} \tag{2.8}$$

Here $*$ denotes an element-wise product. For any layer l, $\mathbf{r}^{(l)}$ is a vector of independent Bernoulli random variables each of which has probability p of being 1. This vector is sampled and multiplied element-wise with the outputs of that layer, $\mathbf{y}^{(l)}$, to create the thinned outputs $\tilde{\mathbf{y}}^{(l)}$. The thinned outputs are then used as input to the next layer. This process is applied at each layer. This amounts to sampling a sub-network from a larger network. For learning, the derivatives of the loss function are backpropagated through the sub-network. At test time, the weights are scaled as $\mathbf{W}_{test}^{(l)} = p\mathbf{W}^{(l)}$. The resulting neural network is used without dropout.

2.4 Calibration

Expected Calibration Error

A model is said to be perfectly calibrated when for each sample $(\mathbf{x}, y) \in \mathcal{D}$, the confidence of the model $\hat{p}$ in the class prediction $\hat{y}$ is equal to the model accuracy $\mathbb{P}(\hat{y} = y|\hat{p})$. For example, of all the data samples that a perfectly calibrated model assigns a prediction confidence of 0.8, 80% of those samples will be predicted correctly. A popular metric used to measure model confidence calibration is the expected calibration error (ECE) [Naeini et al., 2015]. ECE approximates the difference in expectation between model confidence and model accuracy, more formally written as,

$$\mathbb{E}_{\hat{p}}[|\mathbb{P}(\hat{y} = y|\hat{p}) - \hat{p}|]. \tag{2.9}$$

Due to finite data, ECE cannot be computed in practice using (2.9). Instead, we group predictions into M interval bins (each of size $1/M$) and calculate the accuracy and confidence 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 and confidence of B_m are defined as

$$\begin{aligned} \text{acc}(B_m) &= \frac{1}{|B_m|} \sum_{i \in B_m} \mathbb{1}(\hat{y}_i = y_i), \\ \text{conf}(B_m) &= \frac{1}{|B_m|} \sum_{i \in B_m} \hat{p}_i. \end{aligned} \tag{2.10}$$

Where y_i is the true class label for sample i and $\hat{p}_i$ is the model confidence for the class prediction $\hat{y}_i$. ECE is a weighted average of the absolute difference between the accuracy and confidence of each bin,

$$\text{ECE} = \sum_{m=1}^{M} \frac{|B_m|}{N} |\text{acc}(B_m) - \text{conf}(B_m)|. \tag{2.11}$$

A disadvantage of ECE (and all binning-based calibration metrics) is that it is not differentiable and can therefore not be added to the training loss to penalize calibration error [Nixon et al., 2019, Kumar et al., 2018]. Our adaptive framework, introduced in section 4.2, solves this problem and provides a solution for optimizing calibration methods.

Reliability diagrams

A popular visualization method for model calibration that is strongly connected to ECE is reliability diagrams [DeGroot and Fienberg, 1983, Niculescu-Mizil and Caruana, 2005a], which plot the accuracies of confidence bins as a bar chart as illustrated in Figure 2.4. 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, unlike ECE. The advantage of reliability diagrams over ECE is that they can capture model over- and underconfidence. If most of the bars lie below the diagonal, then the model is observed to be overconfident

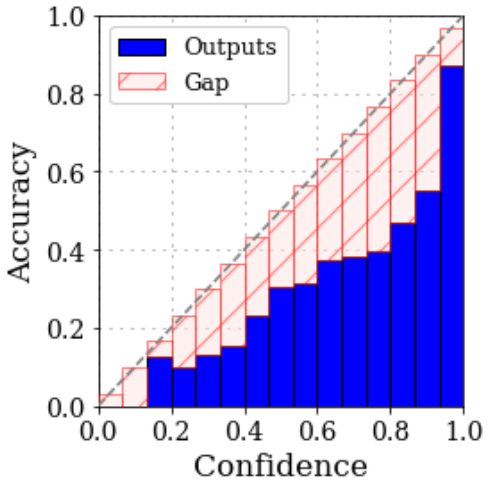


Figure 2.4: Reliability Diagram Example

and similarly with a majority of bars above the diagonal the model is underconfident. This ability to measure and compare model over- and underconfidence is not possible with ECE, and helped motivate our new evaluation metric, confidence balance, defined in section 4.3.

Classwise-ECE

ECE only considers the probability of the predicted class, which means it does not consider how well calibrated a model is with respect to the $K-1$ other probabilities that a model outputs. A stronger definition of calibration requires the probabilities of all the classes for every data instance to be calibrated [Kull et al., 2019, Nixon et al., 2019, Widmann et al., 2019, Kumar et al., 2019, Vaicenavicius et al., 2019]. Classwise-ECE is a simple extension of ECE that accounts for all predictions [Kull et al., 2019, Nixon et al., 2019]. For Classwise-ECE, we group predictions by the K classes and then into M interval bins (each of size $1/M$) and calculate the accuracy and confidence of each bin. Let $B_{k,m}$ be the set of indices of samples from class k whose prediction confidence falls into the interval $I_m = (\frac{m-1}{M}, \frac{m}{M}]$.

The accuracy and confidence of $B_{k,m}$ are defined as

$$\begin{aligned} \text{acc}(B_{k,m}) &= \frac{1}{|B_{k,m}|} \sum_{i \in B_{k,m}} \mathbb{1}(k = y_i), \\ \text{conf}(B_{k,m}) &= \frac{1}{|B_{k,m}|} \sum_{i \in B_{k,m}} \hat{p}_{i,k}. \end{aligned} \tag{2.12}$$

Where $\hat{p}_{i,k}$ is the model confidence that sample i belongs to class k. Classwise-ECE is a weighted average across all K classes and their M bins, of the absolute difference between the bin accuracy and confidence,

$$\text{Classwise-ECE} = \frac{1}{K} \sum_{k=1}^{K} \sum_{m=1}^{M} \frac{|B_{k,m}|}{N} |\text{acc}(B_{k,m}) - \text{conf}(B_{k,m})|. \tag{2.13}$$

Chapter 3

Related Work

Popular statistical learning methods have been studied and analyzed in the context of model calibration. Logistic regression, which has been frequently used successfully in calibration literature [Cox, 1958, Platt, 1999, Guo et al., 2017] produces well calibrated models as it directly optimizes log loss [Elkan, 2007]. Other popular statistical learning methods are less successful at calibration. Naive Bayes models are built on the assumption that features are conditionally independent given the target class. However, features are commonly positively correlated. Zadrozny and Elkan [2001] find that the result of this assumption is that naive Bayes models produce probabilistic estimates close to 0 or 1 which results in poor calibration. Niculescu-Mizil and Caruana [2005b] find maximum margin methods such as Support Vector Machines (with output scaled to [0,1]), boosted trees and boosted stumps suffer from the opposite distortion in their probabilistic estimates, they tend to have risk estimates far away from 0 and 1. They further observe that other types of models such as random forests, bagged trees, and neural nets have been shown to be well calibrated by default in an experimental setting. However, there is no theoretical basis to guarantee these models will be well

calibrated, like with logistic regression. Guo et al. [2017] observed that modern deep neural networks (LSTMs and CNNs) are significantly overconfident (prediction confidence greater than accuracy) and therefore poorly calibrated. In order to correct this overconfidence issue they reviewed and evaluated a number of recalibration methods that rescale model's confidence to be more well calibrated. They reviewed two classes of recalibration methods; parametric and non parametric. The parametric methods, tempurature, vector and matrix scaling were all mutliclass extensions of platt scaling [Platt, 1999] and the non parametric methods were histogram binning [Zadrozny and Elkan, 2001], isotonic regression [Zadrozny and Elkan, 2002] and Bayesian Binning into Quantiles [Naeini et al., 2015] extended to the multiclass setting. Parametric methods proved to be more successful at recalibration, and temperature scaling was the strongest method. Recalibration (post processing calibration) has been a popular area of research in recent years [Kull et al., 2019, Gupta et al., 2021, Flach et al., 2017]. However, Nixon et al. [2019] argue that machine learning models should be calibrated by default, in other words, they should not require any post processing recalibration. In-training calibration methods such as label smoothing, mixup and dropout can produce well calibrated models that do not require recalibration.

In-training calibration methods have been another popular area of research. Common methods for developing new in-training calibration techniques include utilizing new loss functions, building new model architectures or improving baseline methods such as label smoothing. Mukhoti et al. [2020] replace the cross entropy error typically used in deep learning training with focal loss [Lin et al., 2017]. They oberseve that focal loss minimizes a regularised KL divergence between the predicted (softmax) distribution and the target distribution over classes. This results in minimizing the KL divergence while increasing the entropy of the prediction distribution, which prevents the model from becoming overconfi-

dent. The work of Kumar et al. [2018] seeks to minimize an explicit calibration error during training. They introduce Maximum Mean Calibration Error (MMCE) a differentiable proxy for calibration error that they use as a regularization method during training. Xing et al. [2020] develop a dual model system that uses prototypical learning [Snell et al., 2017] to train a classification model and a confidence model simultaneously.

Very little work in confidence calibration has been focused on comparing model over- and underconfidence. Thulasidasan et al. [2019] introduce overconfidence error, a metric which penalizes predictions by the weight of the confidence but only when confidence exceeds accuracy, they use this metric to assess the safety of learning methods for high risk applications. Mund et al. [2015] introduce a definition of over- and underconfidence for their work in active learning, where overconfidence is the average confidence of a classifier on its false predictions and underconfidence is the average uncertainty on its correct predictions. Pleiss et al. [2017] utilize this same definition of over- and underconfidence in an analysis of algorithmic fairness. Under their definition, over- and underconfidence measure confidence and uncertainty respectively on different data instances, they are not quantities that can be compared directly, they are only used for comparison between algorithms as a measure of fairness. Very recently, Wen et al. [2021] introduced a method to improve the calibration performance of an ensemble of models trained with mixup. This work is closely related to our method, as they utilize mixup during training, but if the model becomes underconfident, they switch to standard cross entropy. They do not discuss the possibility of adjusting the mixup parameter as an alternative solution.

3.1 Label Smoothing

Label smoothing, a method that uses soft targets that are a weighted average of hard targets and the uniform distribution over labels was first proposed by Szegedy et al. [2016] as a way to improve the performance of the Inception architecture on the ImageNet dataset. Since then, many state-of-the-art image classification models have utilized label smoothing in their training procedures [Zoph et al., 2018, Real et al., 2019]. Müller et al. [2019] demonstrated the significant improvement in model calibration when trained with label smoothing as opposed to hard targets. Since then, label smoothing has become a popular calibration benchmark, however it is not always accurately reported. Xing et al. [2020] utilize label smoothing as a calibration benchmark to evaluate their dual model, however they do not discuss the effect of the smoothing factor parameter on calibration and they do not even report the value of the smoothing factor used, making the benchmark difficult to reproduce. Other works that use label smoothing as a calibration benchmark [Mukhoti et al., 2020, Thulasidasan et al., 2019, Lukasik et al., 2020] very briefly discuss validating the smoothing factor, however it is generally minimal with at most three values considered and the chosen smoothing factor is then used across all experiments (model architectures, datasets, training setups). Notably the results of label smoothing from Mukhoti et al. [2020] are particularly poor (Table 1.1). Recent work has begun to build off of the uniform label smoothing method to obtain further calibration gains on a variety of datasets and tasks. Krothapalli and Abbott [2020] proposed an adaptive label smoothing method for object detection, where the smoothing factor is updated based on relative object size within an image, in an effort to produce confidences that do not rely on the context of images. Zhang et al. [2020] propose an online label smoothing strategy for image classification that implicitly measures class similarity and

generates soft labels based on the statistics of the model prediction for the target category. In contrast, Liu and JaJa [2020], calculates class similarity explicitly prior to training and uses these scores to obtain smoothing factors for each target class. Of these advances in label smoothing only Liu and JaJa [2020] discuss the effect of the smoothing factor on uniform label smoothing.

3.2 Mixup

Mixup, a data augmentation method for training deep neural networks where additional data samples are created during training by convexly combining pairs of images and their labels was proposed by Zhang et al. [2018]. Although simple to implement, mixup was shown to be very effective, improving the the generalization of state-of-the-art neural network architectures, reducing the memorization of corrupt labels and increasing the robustness to adversarial examples. Thulasidasan et al. [2019] demonstrated that deep neural networks trained with mixup are significanly better calibrated than deep neural networks trained with only regular data augmentation. Since, mixup has become the standard benchmark in calibration research that focuses on data augmentation methods. Similarly to label smoothing Xing et al. [2020] utilize mixup as a calibration benchmark to evaluate their methods, however they do not disclose the parameter value used for their implementation of mixup. Carratino et al. [2020] attempt to explain the theoretical foundations of mixup, they utilize mixup as a baseline in calibration experiments and compare with empirical risk minimization and simple extensions of both methods, they also observe the effect of the mixup parameter value on calibration error. Wen et al. [2021] discuss how data augmentation methods such as mixup do not always improve model calibration when used to train ensemble mod-

els. They present a simple adaptive method to prevent the underconfidence issue caused by combining mixup and ensembles, this adaptive method helps improve calibration over other methods that use ensembles and data augmentations methods together. Mixup has lead to the creation of many other data augmentation methods, the most notable being augmix [Hendrycks et al., 2020], which obtains similar improvements on robustness and calibration measures compared to more traditional methods.

3.3 Dropout

Dropout, a method which randomly drops units and their connections from neural networks during training was originally introduced by Srivastava et al. [2014] as a method to prevent neural network models from overfitting. The method is very effective and is frequently used in applications of neural networks. Gal and Ghahramani [2016] showed that dropout can be viewed as a Bayesian approximation method for representing model uncertainty. Bayesian neural networks can achieve state of the art results for estimating predictive uncertainty [Lakshminarayanan et al., 2017], however they require modifications to the training setup and are computationally expensive (compared to standard non-Bayesian neural networks). Gal and Ghahramani [2016] interpret dropout as a way to create an ensemble model, they sample multiple dropout masks at test time and average the predictions as a way to represent model uncertainty. Havasi et al. [2021] and Wen et al. [2021] utilize dropout as a calibration benchmark in their work on deep neural network ensembles. Ovadia et al. [2019] use dropout as well as last-layer dropout (dropout only applied to the final network layer) as benchmarks for model calibration under dataset shift. Riquelme et al. [2018] introduced the idea of approximate Bayesian inference for the parameters of the last layer only of a neural network.

The results from Ovadia et al. [2019] show that last-layer dropout achieves nearly equal calibration results as more complex dropout schemes on the same model architecture. In all of these works the dropout parameter is not validated.

Chapter 4

Methods

In this chapter, we present our theoretical contributions to this thesis. We provide a decomposition of ECE, into two new metrics, which measure over- and underconfidence calibration error. These new metrics provide the basis for our adaptive parameter framework for model calibration. We present this framework and its assumptions in a general setting, before explaining how label smoothing, mixup and last-layer dropout satisfy the assumptions and can be utilized with our framework. We outline how this framework can be extended with label smoothing to work when each class has its own smoothing factor. Finally, we introduce a new evaluation metric, confidence balance, which measures how well a model is balancing over- and underconfidence calibration error.

4.1 Over- and Underconfidence Metrics

A key feature of reliability diagrams is the ability to analyze model calibration, specifically, over- and underconfidence. The Expected Calibration Error (ECE) metric is closely related to reliability diagrams in that it is a weighted average of the bins' accuracy/confidence

difference. As we have mentioned before, the downside to ECE is that unlike reliability diagrams it does not capture any information about whether a model is over- or underconfident. We present a simple decomposition of ECE (and the notion of miscalibration that it approximates) that provides measures of model over- and underconfidence,

$$\begin{aligned}\text{ECE} &\approx \mathbb{E}_{\hat{p}}[|\mathbb{P}(\hat{y} = y|\hat{p}) - \hat{p}|] \\ &= \underbrace{\mathbb{E}_{\hat{p}}[\mathbb{P}(\hat{y} = y|\hat{p}) - \hat{p} \mid \hat{p} < \mathbb{P}(\hat{y} = y|\hat{p})]}_{\text{underconfidence}} + \underbrace{\mathbb{E}_{\hat{p}}[\hat{p} - \mathbb{P}(\hat{y} = y|\hat{p}) \mid \hat{p} > \mathbb{P}(\hat{y} = y|\hat{p})]}_{\text{overconfidence}}.\end{aligned} \tag{4.1}$$

By conditioning on whether the model accuracy is greater than or less than the model confidence we are able to break up ECE into two terms, one which measures the amount of calibration error that is caused by model underconfidence and the other measures the amount of calibration error that is caused by model overconfidence. In practice the decomposition becomes,

$$\begin{aligned}\text{ECE} &= \sum_{m=1}^{M} \frac{|B_m|}{N} \Big|\text{acc}(B_m) - \text{conf}(B_m)\Big| \\ &= \underbrace{\sum_{m=1}^{M} \frac{|B_m|}{n} \max\left[acc(B_m) - \text{conf}(B_m), 0\right]}_{\text{underconfidence}} \\ &\qquad + \underbrace{\sum_{m=1}^{M} \frac{|B_m|}{n} \max\left[\text{conf}(B_m) - acc(B_m), 0\right]}_{\text{overconfidence}}.\end{aligned} \tag{4.2}$$

Denoting the underconfidence term ECEUC and the overconfidence term ECEOC we have,

$$\text{ECE} = \text{ECEUC} + \text{ECEOC}. \tag{4.3}$$

ECEOC only counts calibration error for confidence bins where the model's average confidence is greater than its' accuracy, and similarly ECEUC only counts calibration error in bins where the model's average confidence is less than model accuracy. These two metrics help recover the information that is lost when using ECE instead of reliablity diagrams, specifically, observing model over- and underconfidence. A perfectly calibrated model will have ECE = 0, which implies ECEOC = ECEUC. In practice, achieving perfect calibration is not possible [Guo et al., 2017], therefore ECE > 0, in this case if ECEOC = ECEUC the model is balanced, however it is not perfectly calibrated. We introduce a new metric (confidence balance) in section 4.3, that measures this concept of calibration balance.

4.2 Adaptive Calibration

Recent work has argued that machine learning models should be calibrated by default [Nixon et al., 2019], and should not require any post processing recalibration such as temperature scaling [Guo et al., 2017]. In the related work section above we have discussed how there are simple methods such as label smoothing, mixup and dropout that, improve model calibration when compared to standard training practices, do not require post processing recalibration and are frequently used as benchmarks to help evaluate more novel calibration methods. However, it is clear that validating the parameter(s) that define these methods is critical for them to perform well and produce a model that has a low calibration error and additionally balances over- and underconfidence. Although empirical validation is sometimes possible, this process is time consuming and should (in theory) be redone whenever there is any change in dataset, model architecture or training procedure. We propose an adaptive calibration method, that can adjust and validate the parameters of these calibration methods during

training by seeking to balance model over- and underconfidence. Using the decomposition of ECE presented above, we are able to define a learning rule to adapt the calibration parameters during training,

$$\theta_{j+1} = \theta_j + \beta(\text{ECEOC}_j - \text{ECEUC}_j). \tag{4.4}$$

Where $\theta_j \geq 0$ is the calibration parameter(s) during training epoch j, $\beta \geq 0$ is the step size of the parameter update, ECEOC_j and ECEUC_j are the over- and underconfidence expected calibration errors respectively, computed on the validation set at the end of epoch j. There are two fundamental assumptions of this adaptive framework. The first assumes that $\theta_j = 0$ represents the calibration method not being used (e.g. smoothing factor $\alpha = 0$, implies learning is done with hard targets). The second assumption is that increasing θ_j will decrease the model's average confidence (e.g. increasing the smoothing factor α, reduces the entropy of the label distribution and decreases the model's confidence). Under these assumptions, if the model becomes overconfident ($\text{ECEOC}_j \geq \text{ECEUC}_j$), θ_j will increase and reduce the model's confidence, where as if the model is underconfident ($\text{ECEOC}_j \leq \text{ECEUC}_j$), θ_j will decrease and the model's average confidence will increase. Since ECE is not differentiable and can therefore not be incorporated into a loss function [Nixon et al., 2019, Kumar et al., 2018], this adaptive framework provides a solution for optimizing calibration methods using only a simple decomposition of ECE.

4.2.1 Adaptive Label Smoothing

For label smoothing we have $\theta = \alpha$, where α is the smoothing factor. $\alpha = 0$ is equivalent to training with no label smoothing (hard targets) thus satisfying the first condition of the

adaptive framework. Müller et al. [2019] demonstrate that label smoothing encourages the differences between the logits of the correct class and incorrect classes to be a constant dependent on α. Therefore, increasing α decreases the difference in logits and lowers the model's prediction confidence, satisfying the second condition of the adaptive framework. Under the adaptive framework, the smoothing factor α is adjusted at the end of every training epoch according to this equation,

$$\alpha_{j+1} = \alpha_j + \beta(\text{ECEOC}_j - \text{ECEUC}_j). \tag{4.5}$$

Where $\alpha_j \in [0, 1]$ is the smoothing factor during epoch j, β is the step size of the smoothing factor update and, ECEOC_j and ECEUC_j are the over- and underconfidence expected calibration errors respectively, computed on the validation set in epoch j. Under this adaptive label smoothing scheme the smoothed targets from standard label smoothing ($q_{i,y}^{LS}$) become indexed by epoch j.

$$q_{i,y,j}^{LS} = (1 - \alpha_j)q_{i,y} + \alpha_j/K. \tag{4.6}$$

4.2.2 Adaptive Mixup

For mixup we have $\theta = \epsilon$, where ϵ is the mixup hyperparameter that controls the strength of the interpolation between pairs of images and the associated smoothing of the training labels. Thulasidasan et al. [2019] discuss that, $\epsilon = 0$ recovers the base case corresponding to zero-entropy training labels, satisfying the first condition of the adaptive framework. They further mention that high values of ϵ result in uniformly averaging the inputs and labels, which corresponds to the case of maximum prediction uncertainty and lower model confidence, satisfying the second condition of the adaptive framework. Under the adaptive

framework, the mixup parameter ϵ is adjusted at the end of every training epoch according to this equation,

$$\epsilon_{j+1} = \epsilon_j + \beta(\text{ECEOC}_j - \text{ECEUC}_j). \tag{4.7}$$

Where $\epsilon_j \in [0, \infty]$ is the mixup parameter during epoch j. Under this adaptive mixup scheme the the linear interpolator $\lambda \in [0, 1]$ that determines the mixing ratio is now drawn from a symmetric Beta distribution that is indexed by epoch j,

$$\lambda_j \sim \text{Beta}(\epsilon_j, \epsilon_j). \tag{4.8}$$

4.2.3 Adaptive Last Layer Dropout

For our experiments with dropout, we utilize last-layer dropout, where dropout is only applied to the last layer in the neural network. last-layer dropout requires only one parameter, the dropout rate p of the last layer. This decision makes the comparison between dropout, mixup and label smoothing more fair since mixup and label smoothing both only utilize one parameter for their methods. For last layer dropout we have $\theta = p$, where p is the dropout rate for the last layer of the neural network. Trivially, $p = 0$ is equivalent to no dropout occurring, which satisfies the first condition of the adaptive framework. Increasing p results in more nodes being dropped, which increases the uncertainty in the model's prediction confidence, satisfying the second condition of the adaptive framework. Under the adaptive framework, the dropout rate p is adjusted at the end of every training epoch according to this equation,

$$p_{j+1} = p_j + \beta(\text{ECEOC}_j - \text{ECEUC}_j). \tag{4.9}$$

Where $p_j \in [0, 1]$ is the dropout rate during epoch j. Under this adaptive last layer dropout scheme the vector of independent Bernoulli random variables $\mathbf{r}_j^{(l)}$ becomes indexed by epoch j.

$$\mathbf{r}_j^{(l)} \sim \text{Bernoulli}(p_j). \tag{4.10}$$

4.2.4 Extensions

Class Adaptive Label Smoothing

Training deep neural networks for image classification, specifically with cross entropy loss has been found to be class-biased [Wang et al., 2019]. Some classes are "easy" for a model to learn and converge faster than other "harder" classes. This results in some classes having significantly higher test accuracy than other classes. This difference in class accuracy motivates the need for model confidence to be adjusted for each class individually, rather than all the classes together. We can extend our adaptive framework to adjust a smoothing factor for each class independently.

$$\alpha_{j+1,k} = \alpha_{j,k} + \beta(\text{ECEOC}_{j,k} - \text{ECEUC}_{j,k}). \tag{4.11}$$

Where $\alpha_{j,k} \in [0, 1]$ is the smoothing factor for class k during epoch j. $\text{ECEOC}_{j,k}$ and $\text{ECEUC}_{j,k}$ are computed using only prediction confidences for class k during epoch j. Here the smoothed targets from standard label smoothing ($q_{i,y}^{LS}$) become indexed by epoch j and class k.

$$q_{i,y,j,k}^{LS} = (1 - \alpha_{j,k})q_{i,y} + \alpha_{j,k}/K. \tag{4.12}$$

4.3 Confidence Balance

Ideally, a model will balance over- and underconfidence, specifically ECEOC = ECEUC, as well as minimize total calibration error. We introduce the Confidence Balance (CB) metric as a way to measure how well a model is balancing over- and underconfidence.

$$\text{CB} = \frac{min\{\text{ECEOC}, \text{ECEUC}\}}{max\{\text{ECEOC}, \text{ECEUC}\}}. \tag{4.13}$$

The value of CB will always be in $[0, 1]$, with CB = 0 representing a model that is purely over- or underconfident with its class predictions and CB = 1 representing a model that is balancing over and underconfidence perfectly. CB is independent of the magnitude of the calibration error. Therefore CB should only be used in addition to calibration error metrics (e.g. ECE, classwise-ECE) during analysis, in order to provide a more complete evaluation of model calibration.

Chapter 5

Results

5.1 Experimental Details

We conduct image classification experiments using the CIFAR-10/100 datasets [Krizhevsky, 2009]. We use a train/validation/test split of 45,000/5,000/10,000 images for both CIFAR-10 and CIFAR-100. We train and evaluate our methods with the VGG11 [Simonyan and Zisserman, 2015] and ResNet50 [He et al., 2016] architectures. The mini-batch size is 128 and we train for 150 epochs with stochastic gradient descent with Nesterov momentum of 0.9, starting with learning rate 0.1 and dropping by a factor of 10 at 75 epochs and 115 epochs. Experiments were run with a NVIDIA Tesla P4 GPU on a cloud provider, training the VGG11 model for 150 epochs took approximately two hours, training the ResNet50 model for 150 epochs took approximately five hours.

We provide two initial benchmarks for comparison; the standard training procedure where our prediction probabilites are simply the softmax values of the model output and the other is the post processing calibration method temperature scaling [Guo et al., 2017]. For the

other benchmarking experiments we use the most commonly used parameter values for the three methods (label smoothing, mixup and last-layer dropout). For label smoothing the smoothing factor is fixed at $\alpha = 0.05$, this is the value most commonly used for CIFAR datasets [Müller et al., 2019, Mukhoti et al., 2020]. For mixup we set $\epsilon = 0.2$, where ϵ parameterizes the beta distribution that is sampled from, this value is used by Zhang et al. [2018] and Thulasidasan et al. [2019]. Finally, for last-layer dropout we fix the dropout rate at 0.2, similarly to the work of Ovadia et al. [2019] and Wen et al. [2021]. For the adaptive experiments we initialize the parameters with the same values as above. We keep these parameter values fixed for the first 30 epochs, this avoids any issue with unnecessary parameter changes early in the training process which could lead to poor model performance. For all adaptive methods the step size is set to $\beta = 0.5$, ECEOC and ECEUC are calculated with 15 bins. We evaluate all methods using model accuracy, ECE (15 bins), classwise-ECE (100 bins), confidence balance, negative log-likelihood, and brier score [Bröcker, 2009].

5.2 Discussion

Generally all three methods achieved better classification accuracy than standard training on the test sets of CIFAR-10 and CIFAR-100 (Table 5.1), agreeing with the original findings of these methods [Szegedy et al., 2016, Zhang et al., 2018, Srivastava et al., 2014]. Between the adaptive framework and non-adaptive method, mixup achieves the best test set accuracy on all four dataset and model architecture combinations. Label smoothing (adaptive and non-adaptive) is very similar in accuracy, and is at most 0.4% behind mixup. Last-layer dropout is less competitive with respect to accuracy and three of the four CIFAR-10 last-layer dropout experiments actually obtained worse accuracy than the standard training procedure. An

Table 5.1: Accuracy (%)

Method	**CIFAR10**		**CIFAR100**	
	VGG11	ResNet50	VGG11	ResNet50
Standard Training	90.10	93.58	66.24	71.91
Temperature Scaling	90.17	93.62	66.22	72.10
Label Smoothing	90.65	93.92	67.59	73.48
Mixup	**90.90**	93.91	**68.00**	73.46
LL Dropout	90.32	92.82	67.84	73.24
Adaptive Label Smoothing (Ours)	90.18	93.91	67.07	73.33
Adaptive Mixup (Ours)	90.62	**93.94**	67.58	**73.77**
Adaptive LL Dropout (Ours)	89.93	92.63	66.93	73.60

observation of the adaptive methods is that their accuracy appears to be slightly lower than the same methods trained without the adaptive framework, this decrease in accuracy is minimal.

The standard training method obtains poor expected calibration error (ECE) in all settings (Table 5.2), agreeing with the findings of Guo et al. [2017]. Both label smoothing and mixup with fixed parameter values of 0.05 and 0.2 respectively are able to significantly reduce ECE compared to the standard training method, this effect has been observed before [Müller et al., 2019, Thulasidasan et al., 2019], this improvement is most notable for CIFAR-100 experiments. Last layer dropout with a fixed parameter value $p = 0.2$ does not improve ECE at all over the standard training procedure for either dataset, agreeing with the findings of Ovadia et al. [2019]. Our adaptive calibration framework performs very well, lowering ECE for all methods, with the only expception being VGG11 with label smoothing for CIFAR-10 where the default parameter used typically in research, happened to be virtually equal to the value our adaptive framework converged too. Since we were using the decomposition

Table 5.2: Expected Calibration Error (ECE)(%)

Method	**CIFAR10**		**CIFAR100**	
	VGG11	ResNet50	VGG11	ResNet50
Standard Training	6.92	4.02	19.07	15.17
Temperature Scaling	3.98	2.86	11.74	5.74
Label Smoothing	4.10	3.96	3.27	5.61
Mixup	7.31	5.24	3.46	3.03
LL Dropout	6.70	4.54	19.43	15.02
Adaptive Label Smoothing (Ours)	4.65	2.83	1.96	2.15
Adaptive Mixup (Ours)	**2.03**	**2.50**	**1.95**	**1.51**
Adaptive LL Dropout (Ours)	2.38	2.89	6.15	6.98

of ECE to adjust our parameters this improvement in ECE is expected. Adaptive mixup performs the best, achieving the lowest ECE on all four experimental setups. Adaptive label smoothing is competitive as well. Adaptive last-layer dropout achieves a great reduction in ECE compared to its regular version with fixed parameter, however even with the improvement it is still beaten by mixup and label smoothing with fixed default parameter values. For top-1 classification calibration, last layer dropout is a poor choice for a benchmark, label smoothing and mixup are far better options.

As expected the confidence balance for non-adaptive methods was poor (Table 5.3). The only instances of non-adaptive methods achieving a reasonable confidence balance was when the default parameter used in those experiments was roughly optimal (Table 5.6). The standard training models and non-adaptive dropout models were overconfident at the end of training where as the non-adaptive mixup and label smoothing models were generally underconfident. This poor balance between over- and underconfidence with trained models motivates our adaptive framework. It can become very costly to rerun these experiments

Table 5.3: Confidence Balance (CB)(%)

Method	**CIFAR10**		**CIFAR100**	
	VGG11	ResNet50	VGG11	ResNet50
Standard Training	0.05	0.53	0	0
Temperature Scaling	0	0	0	0
Label Smoothing	90.86	28.49	0.15	0.08
Mixup	2.75	2.18	0	49.35
LL Dropout	0	0	0	0
Adaptive Label Smoothing (Ours)	**91.70**	**96.58**	59.60	23.56
Adaptive Mixup (Ours)	6.87	45.89	0.03	**87.24**
Adaptive LL Dropout (Ours)	66.57	1.48	**97.79**	77.04

until you find a suitable parameter value for optimal calibration, our adaptive framework removes this need for external parameter validation and instead finds a good parameter value during training. From Table 5.3, we see that the adaptive methods were much more successful at balancing over- and underconfidence on the test data. Label smoothing achieves the best results and obtains a good confidence balance in all four experimental settings, last layer dropout worked well in three settings and adaptive mixup was the worst performing with only two of the four resulting models obtaining a good confidence balance. Label smoothing performing the best with respect to confidence balance is most likely due to its close relation to the cross entropy loss function we are optimizing. With label smoothing, when the smoothing factor is changed during training this effect is immediately applied to the loss functions and every mini batch of data in the next training epoch, therefore label smoothing reacts very precisely and quickly to the changes made by the adaptive framework. With dropout when the parameter p is changed this effect is immediate as well and is applied to the next training epoch, however the difference with dropout is that the nodes that are

Table 5.4: Classwise-ECE(%)

Method	**CIFAR10**		**CIFAR100**	
	VGG11	ResNet50	VGG11	ResNet50
Standard Training	1.54	0.95	0.52	0.43
Temperature Scaling	1.04	0.92	0.40	0.34
Label Smoothing	1.18	1.15	0.38	0.38
Mixup	1.99	1.41	0.38	0.35
LL Dropout	1.50	1.03	0.51	0.45
Adaptive Label Smoothing (Ours)	1.23	0.90	0.37	0.34
Adaptive Mixup (Ours)	**0.90**	0.83	0.37	0.32
Adaptive LL Dropout (Ours)	1.00	**0.79**	**0.32**	**0.28**

removed are different for every data instance, therefore this change is not uniformly applied across the training data and slows down the response from the parameter changes. Mixup suffers from a similar problem to dropout, a delay between the calibration parameter being changed by the adaptive framework and the validation statistics reflecting this change. For mixup this is caused by a disconnect between the mixup parameter ϵ that is altered during training and the parameter λ used to create the convex combinations of data instances. The mixup parameter ϵ only defines the distribution from which λ is drawn from, therefore this sampling procedure delays the response between changing the value of ϵ and the validation statistics reflecting that change. Label smoothing is most suited for the adaptive framework as the changes in the smoothing factor α are immediately and directly applied to the loss function and the change in α results in a uniform change for all training data labels.

The classwise-ECE results (Table 5.4) further demonstrate how the adaptive framework produces more well calibrated models than standard methods. Across all three benchmark methods, using adaptive parameter tuning produces lower classwise-ECE. The most interest-

Table 5.5: Negative Log-Likelihood (NLL)

Method	**CIFAR10**		**CIFAR100**	
	VGG11	ResNet50	VGG11	ResNet50
Standard Training	0.49	0.28	1.74	1.31
Temperature Scaling	0.33	0.22	1.61	1.11
Label Smoothing	0.36	0.24	1.38	1.14
Mixup	0.36	0.24	1.29	1.11
LL Dropout	0.46	0.30	1.75	1.31
Adaptive Label Smoothing (Ours)	0.37	0.24	1.38	1.10
Adaptive Mixup (Ours)	**0.32**	**0.21**	**1.29**	**1.02**
Adaptive LL Dropout (Ours)	0.42	0.29	1.45	1.15

ing results of Table 5.4 is that adaptive last layer dropout achieves the lowest classwise-ECE on three of the four experimental setups. This is contrary to the results of ECE (Table 5.2), where adaptive last-layer dropout performed poorly. Although adaptive last-layer dropout is relatively poor at top-1 calibration (ECE) it is able to provide accurate confidence measures for non class predictions (Classwise-ECE).

Our adaptive framework appears to decrease negative log likelihood (NLL) loss (Table 5.5). This improvement in NLL makes sense as it has previously been proven that NLL can be decomposed into three components; uncertainty, resolution and calibration [Bröcker, 2009]. Therefore since Table 5.2 and 5.4 demonstrate that calibration error decreases with our adaptive methods, it follows that NLL will decrease as well, assuming uncertainty and resolution remain constant. We observe that similar to ECE, adaptive mixup achieves the lowest NLL on all four experimental setups.

Our adaptive framework appears to decrease Brier score (Table 5.6). This improvement in Brier score makes sense as it has previously been proven that Brier score can be decomposed

Table 5.6: Brier Score

Method	**CIFAR10**		**CIFAR100**	
	VGG11	ResNet50	VGG11	ResNet50
Standard Training	7.99	4.94	20.01	16.73
Temperature Scaling	6.98	4.04	16.54	13.46
Label Smoothing	6.70	3.65	15.43	12.09
Mixup	6.51	3.47	14.90	11.83
LL Dropout	7.73	4.79	19.66	16.21
Adaptive Label Smoothing (Ours)	6.65	3.42	14.87	11.63
Adaptive Mixup (Ours)	**6.39**	**3.27**	**14.38**	**11.55**
Adaptive LL Dropout (Ours)	7.03	4.34	15.07	12.01

into two components; calibration and sharpness [Bröcker, 2009]. Therefore since Table 5.2 and 5.4 demonstrate that calibration error decreases with our adaptive methods, it follows that Brier score will decrease as well. We observe that similar to ECE, adaptive mixup achieves the lowest Brier score on all four experimental setups.

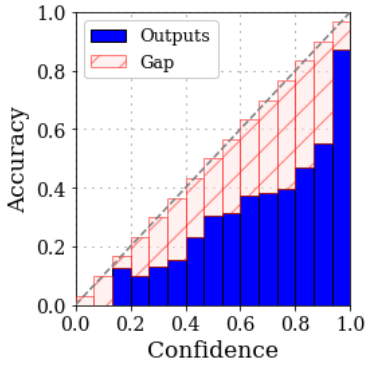
Finally we examine how the parameter values evolved over training (Table 5.7). For label smoothing, the smoothing factor did depend on the dataset and model architecture, the smoothing factor at the end of training for VGG11 on CIFAR-10 was the highest and ResNet50 on CIFAR-100 was the lowest. Therefore for label smoothing, larger model architectures and datasets with more classes require smaller smoothing factors. An important observation though is that the difference between the final smoothing factor value between all four experiments was only 0.03, therefore there is not significant variation in optimal smoothing factor between datasets and model architectures. Mixup does not appear to have this same property, there is a large difference between the final mixup parameter values of CIFAR-10 and CIFAR-100 experiments. For the same dataset, the difference in final

Table 5.7: Calibration parameter analysis: initial parameter values (θ_0) compared to parameter values at the end of training for adaptive methods across datasets and model architectures

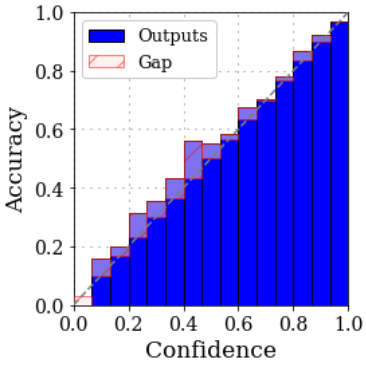
Method	$\boldsymbol{\theta_0}$	**CIFAR10**		**CIFAR100**	
		VGG11	ResNet50	VGG11	ResNet50
Adaptive Label Smoothing	0.05	0.053	0.033	0.036	0.027
Adaptive Mixup	0.2	0.100	0.097	0.186	0.177
Adaptive LL Dropout	0.2	0.966	0.988	0.962	0.979

parameter value is minimal much like label smoothing. Therefore we see that selecting a good mixup parameter is very dependent on the dataset. The dropout parameter finishing above 0.95 is very surprising, considering most work with last layer dropout sets the parameter at 0.2. We observe that in order to balance over and underconfidence the dropout rate increases significantly. An interesting observation is that obtaining a good confidence balance and calibration error with last layer dropout is only possible with our method. We have demonstrated that a fixed value of 0.2 keeps the model overconfident, but if we use a dropout rate of 0.95 throughout training, learning is too slow. Therefore obtaining a well calibrated model with last layer dropout is only possible with our adaptive framework that slowly enables the dropout rate to increase during training in order to balance model over- and underconfidence.

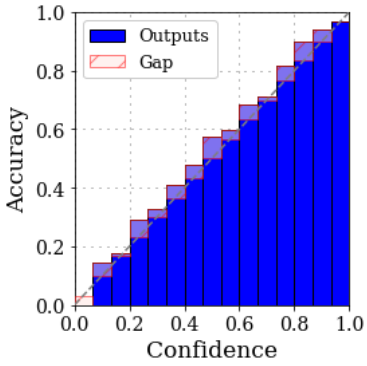
Figure 5.1 presents the reliability diagrams for all seven methods evaluated on one of the four dataset and model architecture experimental setups. This figure provides a visual analysis to support both our discussion about calibration error metrics (ECE, classwise-ECE) and our new calibration balance metric (CB). The additional reliability diagrams for the other three experimental setups can be found in appendix A. standard training and last-layer dropout (p=0.2) achieve poor model calibration, both models are significantly overconfident.



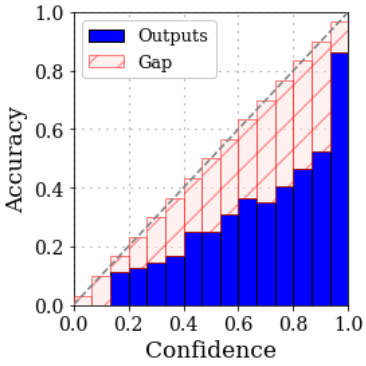
(a) standard training



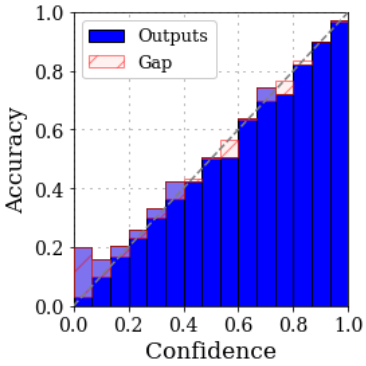
(b) label smoothing



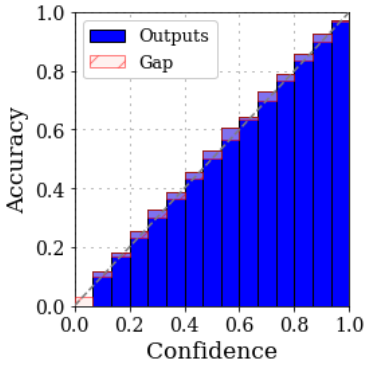
(c) mixup



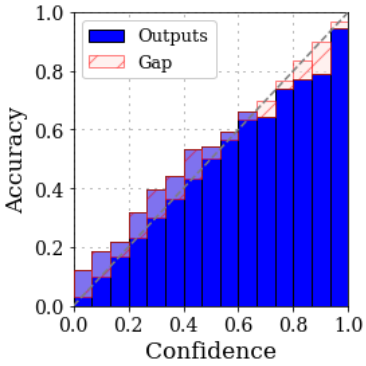
(d) last-layer dropout



(e) adaptive label smoothing



(f) adaptive mixup



(g) adaptive last-layer dropout

Figure 5.1: Reliability Diagrams for VGG11 model trained on CIFAR100

Label smoothing and mixup are more well calibrated, most bins have relativly small error, but a clear trend of underconfidence can be observed in every bin. Adaptive last-layer dropout is significantly more well calibrated than last-layer dropout with a fixed dropout rate, however it still appears worse off than label smoothing and mixup with fixed parameters. Adaptive last-layer dropout displays underconfidence on low confidence predictions and overconfidence on high confidence predictions. This reliability diagram demonstrates why the confidence balance metric should be reported along with calibration error, for this model CB $\approx 98\%$ however the ECE is three times greater than for adaptive label smoothing and mixup. The last-layer dropout reliability diagram also points out a clear weakness in the formulation of confidence balance, the CB of 98% is a near perfect score, however if you were to calculate the confidence balance on the first ten bins and the last 5 bins separately each would have CB $= 0$, this further supports the idea that confidence balance must be reported along with calibration error metrics. Adaptive label smoothing appears to be well calibrated, and appears to have improved slightly from label smoothing with a fixed smoothing factor value. Finally, adaptive mixup is clearly the most well calibrated model, it maintains a trend of underconfidence in each bin, but the bin errors are very small and very consistent across the entire confidence range. Visually, adaptive mixup was clearly the best method for this particular dataset and model combination.

Class Adaptive Label Smoothing

Class adaptive label smoothing improves ECE over regular label smoothing in all experiments and over adaptive label smoothing in three of the four experiments (Table 5.8). The improvement in ECE for class adaptive label smoothing on CIFAR-10 is significant, compared to standard label smoothing and adaptive label smoothing it lowers ECE by at least

Table 5.8: Class adaptive label smoothing analysis

Method	**CIFAR10-VGG11**			**CIFAR10-ResNet50**		
	Accuracy%	ECE%	CB%	Accuracy%	ECE%	CB%
Standard Training	90.10	6.93	0.06	93.58	4.02	0.54
Label Smoothing	**90.65**	4.10	90.87	93.92	3.96	28.49
Adaptive LS (Ours)	90.18	4.65	**91.70**	93.91	2.83	**96.59**
Class Adaptive LS (Ours)	90.24	**2.67**	58.61	**93.98**	**1.72**	80.72
Method	**CIFAR100-VGG11**			**CIFAR100-ResNet50**		
	Accuracy%	ECE%	CB%	Accuracy%	ECE%	CB%
Standard Training	66.24	19.07	0.0	71.91	15.17	0.0
Label Smoothing	67.59	3.27	0.15	**73.48**	5.61	0.08
Adaptive LS (Ours)	67.07	**1.95**	59.60	73.33	2.15	53.55
Class Adaptive LS (Ours)	**67.78**	2.43	**62.33**	73.37	**1.82**	**77.16**

1%, clearly indicating that a smoothing factor dedicated for each different class can have significant benefits on model calibration. Class adaptive label smoothing on CIFAR-100 is less impressive, increasing ECE with the VGG11 architecture and barely decreasing ECE with the ResNet50 architecture. The difference between CIFAR-10 and CIFAR-100 is the number of image classes (10 and 100 respectively), therefore the number of images per class is ten times less in CIFAR-100. In the CIFAR-10 experiments estimates for ECEOC and ECEUC for each image class are computed with approximately 500 images (5000 images total in the validation set), these same estimates for class adaptive label smoothing on CIFAR-100 are computed with approximately 50 images. The decrease in images per class from CIFAR-10 to CIFAR-100 make the confidence estimates that class adaptive label smoothing uses less accurate and more variable, therefore the performance drops significantly. There are a number of possible options to obtain more confidence data for each image class, the top-r

Table 5.9: Class adaptive Classwise-ECE (%)

Method	**CIFAR10**		**CIFAR100**	
	VGG11	ResNet50	VGG11	ResNet50
Standard Training	1.54	1.34	0.52	0.43
Label Smoothing	1.18	1.15	0.38	0.39
Adaptive Label Smoothing (Ours)	1.22	0.90	0.37	0.34
Class Adaptive LS (Ours)	**1.16**	**0.83**	**0.36**	**0.33**

prediction probabilities could be considered, as opposed to the current method of top-1, a similar idea was proposed by Gupta et al. [2021]. Another option would be to use any prediction confidence above a certain threshold, Nixon et al. [2019] use this concept in their calibration evaluation framework. Further calibration evaluation is presented in Table 5.9 with Classwise-ECE. Class adaptive label smoothing had the lowest classwise-ECE for all experiments, demonstrating that adjusting a smoothing factor for each target class has a significant benefit over a joint smoothing factor used for every class.

Chapter 6

Conclusion

In this work we have focused on improving methods for model confidence calibration. We have shown that the most popular calibration methods; label smoothing, mixup and last-layer dropout, are relatively simple in theory and practice, and confirmed previous observations that these methods can improve model calibration compared to using standard training methods. We demonstrate that the parameter values that define these methods can have a significant influence on how well these methods work for model calibration, and that these optimal parameter values can change depending on dataset and model architecture. Research that utilizes these methods rarely validate the parameter values and when they do, their procedures are not extensive, most likely due to the computational cost of training vision models for classification tasks. We introduced a decomposition of expected calibration error (ECE) that provides metrics to measure over- and underconfidence. These two new metrics enabled us to develop a framework for optimizing calibration methods by adjusting method parameters in order to keep the over- and underconfidence metrics relatively equal. This framework was motivated by the lack of attention and discussion on model

over- and underconfidence in recent calibration research and since binning-based calibration metrics are not differentiable and therefore cannot be incorporated into loss functions. This framework also enables the method parameters to be validated during training, alleviating the cost of traditional parameter validation. Using this framework with popular calibration methods (label smoothing, mixup and last-layer dropout) we demonstrate its effectiveness; lower calibration error and a better balance between model over- and underconfidence, while preserving classification accuracy. We extend this idea to a class based adaptive framework, which we test with label smoothing. This extension appears to work well when sufficient data is supplied for each class. Our work also serves as a good comparison between label smoothing, mixup and last-layer dropout for calibration experiments. Our results show that when all three methods are used with our adaptive framework, mixup produces the most well calibrated models, beating label smoothing by a small margin but consistently across experimental setups (datasets and model architectures) and beating last-layer dropout by a significant margin.

6.1 Future Work

There are multiple avenues of interest for future work. Applying our adaptive framework to more recently developed methods for calibration such as focal loss [Mukhoti et al., 2020] could further validate the effectiveness of the framework, and provide a better comparison of how well these new methods compare to more widely used methods such as label smoothing and mixup. We demonstrated the potential for our class based adaptive framework. Future work could explore whether utilizing the top-r prediction probabilities could help resolve the issues we experienced on CIFAR-100 with a lack of data for each class. It would be interesting

to also apply the class adaptive framework to mixup (Wen et al. [2021] have done something similar very recently) and last-layer dropout. Our work focused on the simple case of methods that can be defined by a single parameter. In fact our framework can be viewed as a simple one-dimensional optimization method. It would be interesting to explore extensions of this framework for methods with multiple parameters, such as augmix [Hendrycks et al., 2020] or dropout applied to multiple layers in a neural network. Expanding our framework to multiple parameters could potentially require a more sophisticated optimization method. Despite its popularity, ECE has quite a few weaknesses [Nixon et al., 2019, Kumar et al., 2019]. Since our measures of over- and underconfidence for this work came from the decomposition of ECE, our framework may have suffered slightly either by the number of bins we choose to use for these metrics or simply by the fact that we choose ECE. It would be interesting to do a review of the adaptive framework experiments in this thesis using different numbers of bins for ECEOC and ECEUC to see if it improves the performance at all, or potentially trying another calibration error metric (and its subsequent decomposition) such as classwise-ECE.

Bibliography

Ana Carolina Alba, Thomas Agoritsas, Michael Walsh, Steven Hanna, Alfonso Iorio, P. J. Devereaux, Thomas McGinn, and Gordon Guyatt. Discrimination and Calibration of Clinical Prediction Models: Users' Guides to the Medical Literature. *JAMA*, 318(14): 1377–1384, 2017.

Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.

Jochen Bröcker. Reliability, sufficiency, and the decomposition of proper scores. *Quarterly Journal of the Royal Meteorological Society*, 135(643):1512–1519, 2009.

Luigi Carratino, Moustapha Cisse, Rodolphe Jenatton, and Jean-Philippe Vert. On mixup regularization. *arXiv preprint arXiv:2006.06049*, 2020.

Olivier Chapelle, Jason Weston, Léon Bottou, and Vladimir Vapnik. Vicinal risk minimization. In *Advances in Neural Information Processing Systems*, 2001.

David Cox. Two further applications of a model for binary regression. *Biometrika*, 45: 562–565, 1958.

Morris H. DeGroot and Stephen E. Fienberg. The comparison and evaluation of forecasters. *Journal of the Royal Statistical Society. Series D (The Statistician)*, pages 12–22, 1983.

Charles Elkan. Log-linear models and conditional random fields. 2007.

Peter Flach, Meelis Kull, and Telmo Silva Filho. Beta calibration: a well-founded and easily implemented improvement on logistic calibration for binary classifiers. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, 2017.

Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of the 33rd International Conference on Machine Learning*, 2016.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, 2017.

Kartik Gupta, Amir Rahimi, Thalaiyasingam Ajanthan, Thomas Mensink, Cristian Sminchisescu, and Richard Hartley. Calibration of neural networks using splines. In *International Conference on Learning Representations*, 2021.

Marton Havasi, Rodolphe Jenatton, Stanislav Fort, Jeremiah Liu, Jasper Roland Snoek, Balaji Lakshminarayanan, Andrew Mingbo Dai, and Dustin Tran. Training independent subnetworks for robust prediction. In *International Conference on Learning Representations*, 2021.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

Dan Hendrycks, Norman Mu, Ekin D. Cubuk, Barret Zoph, Justin Gilmer, and Balaji Lakshminarayanan. AugMix: A simple data processing method to improve robustness and uncertainty. *Proceedings of the International Conference on Learning Representations*, 2020.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

Ujwal Krothapalli and A. Lynn Abbott. Adaptive label smoothing. *arXiv preprint arXiv:2009.06432*, 2020.

Meelis Kull, Miquel Perello Nieto, Markus Kängsepp, Telmo Silva Filho, Hao Song, and Peter Flach. Beyond temperature scaling: Obtaining well-calibrated multi-class probabilities with dirichlet calibration. In *Advances in Neural Information Processing Systems*, 2019.

Ananya Kumar, Percy S Liang, and Tengyu Ma. Verified uncertainty calibration. In *Advances in Neural Information Processing Systems*, 2019.

Aviral Kumar, Sunita Sarawagi, and Ujjwal Jain. Trainable calibration measures for neural networks from kernel mean embeddings. In *Proceedings of the 35th International Conference on Machine Learning*, 2018.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, 2017.

Jesse Levinson, Jake Askeland, Jan Becker, Jennifer Dolson, David Held, Soeren Kammel, J. Zico Kolter, Dirk Langer, Oliver Pink, Vaughan Pratt, Michael Sokolsky, Ganymed

Stanek, David Stavens, Alex Teichman, Moritz Werling, and Sebastian Thrun. Towards fully autonomous driving: Systems and algorithms. In *IEEE Intelligent Vehicles Symposium*, 2011.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. Focal loss for dense object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, 2017.

Chihuang Liu and Joseph JaJa. Class-similarity based label smoothing for generalized confidence calibration. *arXiv preprint arXiv:2006.14028*, 2020.

Michal Lukasik, Srinadh Bhojanapalli, Aditya Menon, and Sanjiv Kumar. Does label smoothing mitigate label noise? In *Proceedings of the 37th International Conference on Machine Learning*, 2020.

Riccardo Miotto, Li Li, and Brian Kidd. Deep patient: An unsupervised representation to predict the future of patients from the electronic health records. *Scientific Reports*, 2016.

Jishnu Mukhoti, Viveka Kulharia, Amartya Sanyal, Stuart Golodetz, Philip Torr, and Puneet Dokania. Calibrating deep neural networks using focal loss. In *Advances in Neural Information Processing Systems*, 2020.

Rafael Müller, Simon Kornblith, and Geoffrey E Hinton. When does label smoothing help? In *Advances in Neural Information Processing Systems*, 2019.

Dennis Mund, Rudolph Triebel, and Daniel Cremers. Active online confidence boosting for efficient object classification. In *2015 IEEE International Conference on Robotics and Automation*, 2015.

Mahdi Pakdaman Naeini, Gregory Cooper, and Milos Hauskrecht. Obtaining well calibrated probabilities using bayesian binning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2015.

Alexandru Niculescu-Mizil and Rich Caruana. Predicting good probabilities with supervised learning. In *Proceedings of the 22nd International Conference on Machine Learning*, 2005a.

Alexandru Niculescu-Mizil and Rich Caruana. Predicting good probabilities with supervised learning. In *Proceedings of the 22nd International Conference on Machine Learning*, 2005b.

Jeremy Nixon, Michael W. Dusenberry, Linchuan Zhang, Ghassen Jerfel, and Dustin Tran. Measuring calibration in deep learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2019.

Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, D. Sculley, Sebastian Nowozin, Joshua Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. In *Advances in Neural Information Processing Systems*, 2019.

John C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in Large Margin Classifiers*, 1999.

Geoff Pleiss, Manish Raghavan, Felix Wu, Jon Kleinberg, and Kilian Q Weinberger. On fairness and calibration. In *Advances in Neural Information Processing Systems*, 2017.

Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V. Le. Regularized evolution for image classifier architecture search. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019.

Carlos Riquelme, George Tucker, and Jasper Snoek. Deep bayesian bandits showdown: An empirical comparison of bayesian deep networks for thompson sampling. In *International Conference on Learning Representations*, 2018.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2015.

Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, 2017.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

Niko Sünderhauf, Oliver Brock, Walter Scheirer, Raia Hadsell, Dieter Fox, Jürgen Leitner, Ben Upcroft, Pieter Abbeel, Wolfram Burgard, Michael Milford, and Peter Corke. The limits and potentials of deep learning for robotics. *The International Journal of Robotics Research*, 37(4-5):405–420, 2018.

Sunil Thulasidasan, Gopinath Chennupati, Jeff A Bilmes, Tanmoy Bhattacharya, and Sarah Michalak. On mixup training: Improved calibration and predictive uncertainty for deep neural networks. In *Advances in Neural Information Processing Systems*, 2019.

Juozas Vaicenavicius, David Widmann, Carl Andersson, Fredrik Lindsten, Jacob Roll, and

Thomas Schön. Evaluating model calibration in classification. In *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, 2019.

Yisen Wang, Xingjun Ma, Zaiyi Chen, Yuan Luo, Jinfeng Yi, and James Bailey. Symmetric cross entropy for robust learning with noisy labels. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019.

Yeming Wen, Ghassen Jerfel, Rafael Muller, Michael W Dusenberry, Jasper Snoek, Balaji Lakshminarayanan, and Dustin Tran. Combining ensembles and data augmentation can harm your calibration. In *International Conference on Learning Representations*, 2021.

David Widmann, Fredrik Lindsten, and Dave Zachariah. Calibration tests in multi-class classification: A unifying framework. In *Advances in Neural Information Processing Systems*, 2019.

Chen Xing, Sercan Arik, Zizhao Zhang, and Tomas Pfister. Distance-based learning from errors for confidence calibration. In *International Conference on Learning Representations*, 2020.

Bianca Zadrozny and Charles Elkan. Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers. In *Proceedings of the Eighteenth International Conference on Machine Learning*, 2001.

Bianca Zadrozny and Charles Elkan. Transforming classifier scores into accurate multiclass probability estimates. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2002.

Chang-Bin Zhang, Peng-Tao Jiang, Qibin Hou, Yunchao Wei, Qi Han, Zhen Li, and Ming-Ming Cheng. Delving deep into label smoothing. *arXiv preprint arXiv:2011.12562*, 2020.

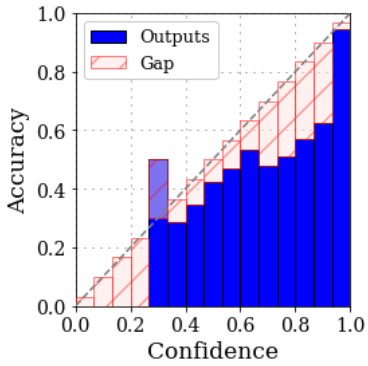
Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018.

Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

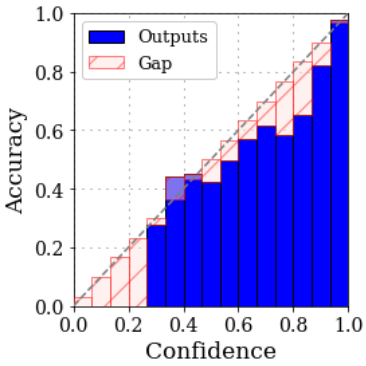
Appendix A

Reliability Diagrams

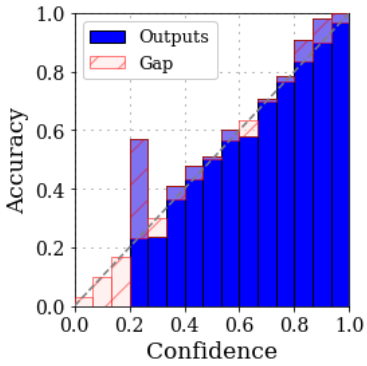
Here, we follow up from Figure 5.1 and provide the reliability diagrams from our other three experiments. Overall the general trends that we discussed with Figure 5.1 hold for these sets of reliability diagrams as well. Most importantly our adaptive method appears to improve calibration across all three methods; label smoothing, mixup and last-layer dropout, this improvement is most noticeable in Figure A.3. Additionally, we observe in these reliability diagrams that our adaptive method works best with mixup, nearly as well with label smoothing and last-layer dropout performs reasonably on CIFAR10 experiments (Figure A.1,2) and poorly on CIFAR100 (Figure A.3).



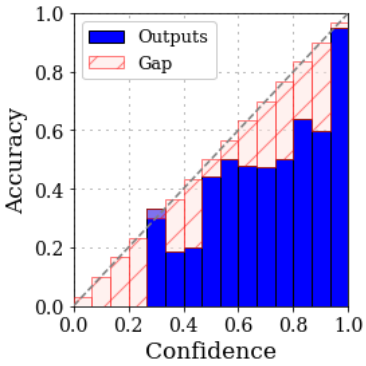
(a) standard training



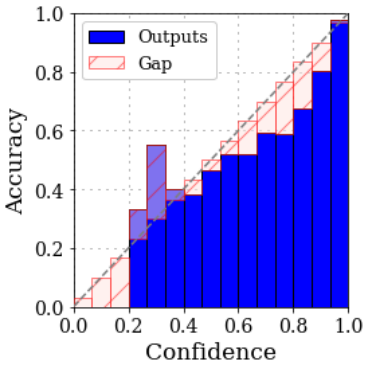
(b) label smoothing



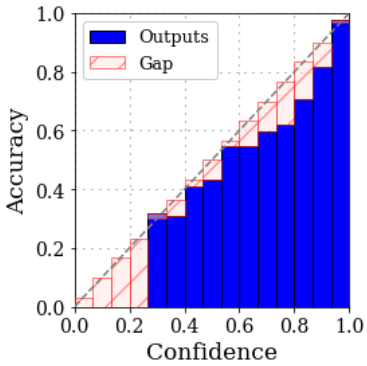
(c) mixup



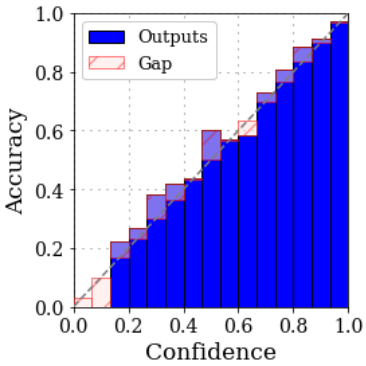
(d) last-layer dropout



(e) adaptive label smoothing

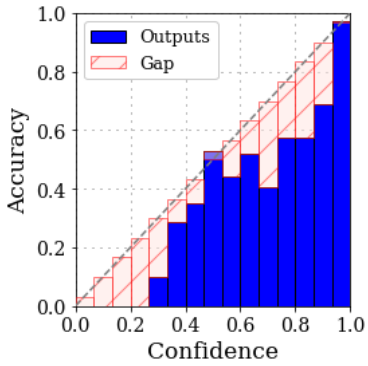


(f) adaptive mixup

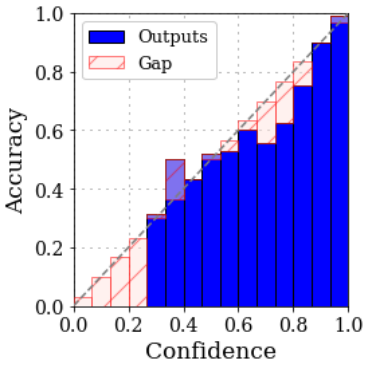


(g) adaptive last-layer dropout

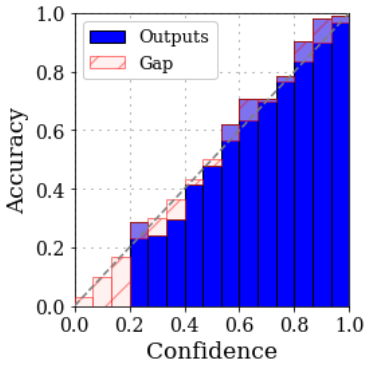
Figure A.1: Reliability Diagrams for VGG11 model trained on CIFAR10



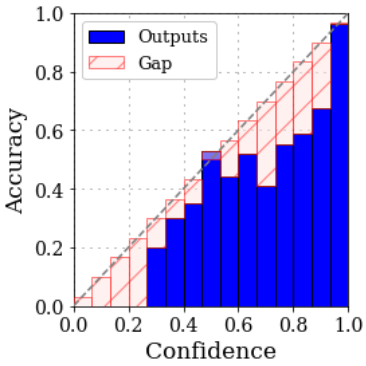
(a) standard training



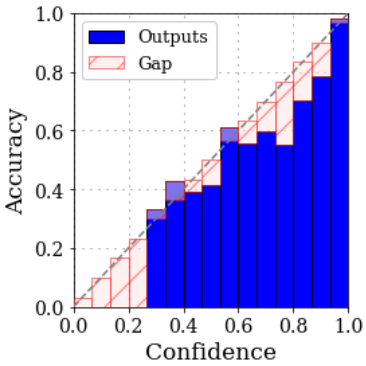
(b) label smoothing



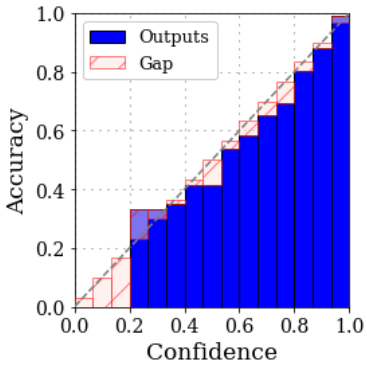
(c) mixup



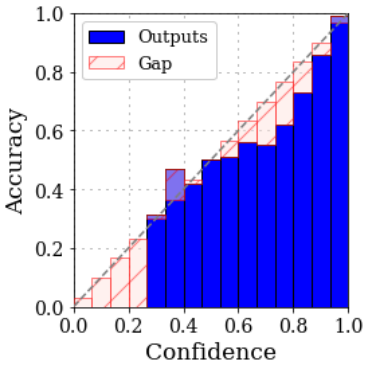
(d) last-layer dropout



(e) adaptive label smoothing

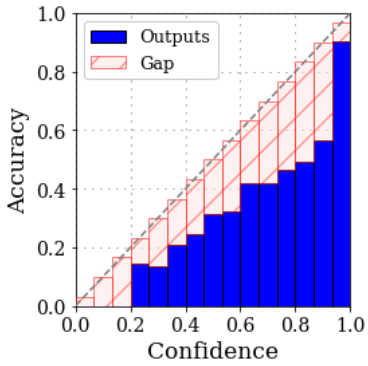


(f) adaptive mixup

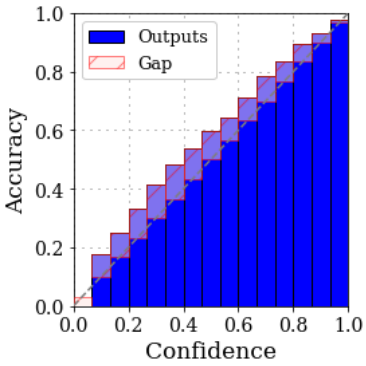


(g) adaptive last-layer dropout

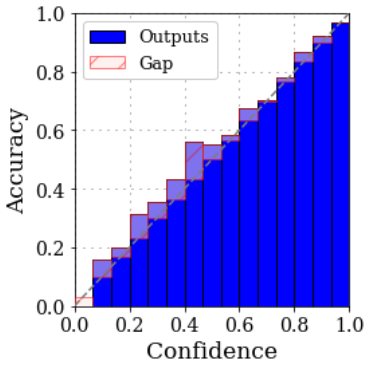
Figure A.2: Reliability Diagrams for ResNet50 model trained on CIFAR10



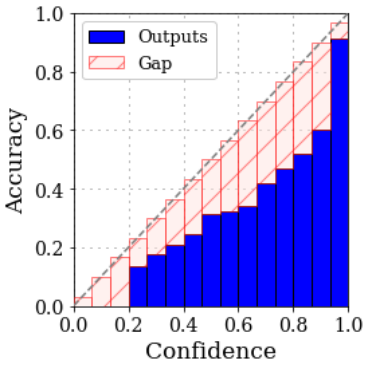
(a) standard training



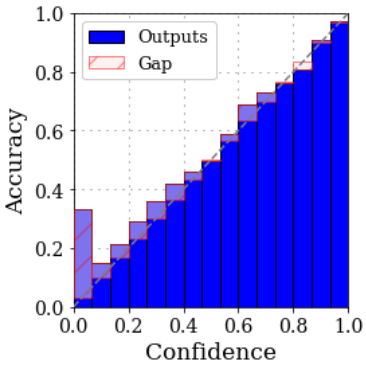
(b) label smoothing



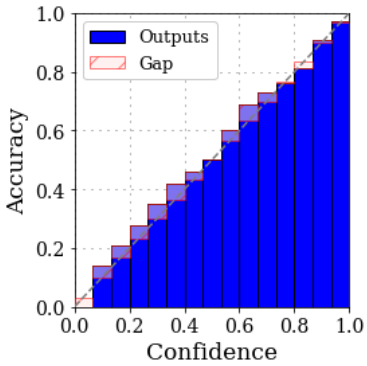
(c) mixup



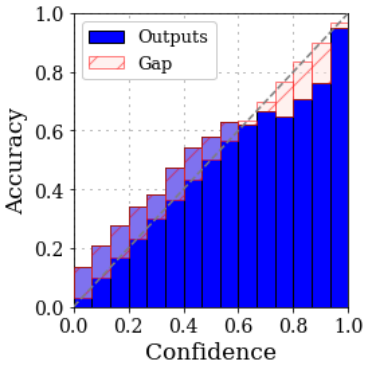
(d) last-layer dropout



(e) adaptive label smoothing



(f) adaptive mixup



(g) adaptive last-layer dropout

Figure A.3: Reliability Diagrams for ResNet50 model trained on CIFAR100