# Calibrating Deep Neural Networks by Pairwise Constraints

Jiacheng Cheng Nuno Vasconcelos
Department of Electrical and Computer Engineering
University of California, San Diego
{jicheng,nvasconcelos}@ucsd.edu

## Abstract

*It is well known that deep neural networks (DNNs) produce poorly calibrated estimates of class-posterior probabilities. We hypothesize that this is due to the limited calibration supervision provided by the cross-entropy loss, which places all emphasis on the probability of the true class and mostly ignores the remaining. We consider how each example can supervise all classes and show that the calibration of a $C$-way classification problem is equivalent to the calibration of $C(C-1)/2$ pairwise binary classification problems that can be derived from it. This suggests the hypothesis that DNN calibration can be improved by providing calibration supervision to all such binary problems. An implementation of this calibration by pairwise constraints (CPC) is then proposed, based on two types of binary calibration constraints. This is finally shown to be implementable with a very minimal increase in the complexity of cross-entropy training. Empirical evaluations of the proposed CPC method across multiple datasets and DNN architectures demonstrate state-of-the-art calibration performance.*

## 1. Introduction

Deep neural networks (DNNs), especially deep convolutional neural networks, have enabled significant advances in computer vision [17,23]. While achieving state-of-the-art accuracy in various tasks such as image recognition [8, 43] and segmentation [25, 41], DNNs do not excel at estimating the confidence of their predictions. Although they output class-posterior probabilities via softmax regression, it is well known that these predictive probabilities are usually poorly calibrated. Frequently, DNNs tend be *overconfident*, assigning high confidence to incorrect predictions [5, 6, 34].

For many real-world applications (e.g. weather forecasting [3, 29, 30], medical diagnosis [13]), it is important that a classifier output not only accurate predictions but also sound estimates of confidence in these predictions. This is known as calibration. For a calibrated classifier, a posterior probability of $p$ for a given class, implies that selecting the class will result in the correct classification $p \times 100\%$ of the time. Consider, for example, a medical diagnosis setting where a diagnostic accuracy above 95% is required for any system to be considered "human equivalent". A diagnostic classifier with accuracy of 80% fails to meet this criterion. However, if the classifier can accurately predict the posterior probabilities associated with its predictions, it can still be useful: Predictions with posterior probabilities above 95% can be accepted automatically, and only the examples of predictions with lower confidence need to be routed to human doctors. Since all the "easy" cases tend to be in the first class, this can reduce the need for human inspection to a relatively small batch of "hard" examples, saving significant time and expense. For these reasons, the probability calibration of DNNs is attracting increasing attention in the computer vision and machine learning communities [18, 19, 21, 27, 35, 48, 52, 56].

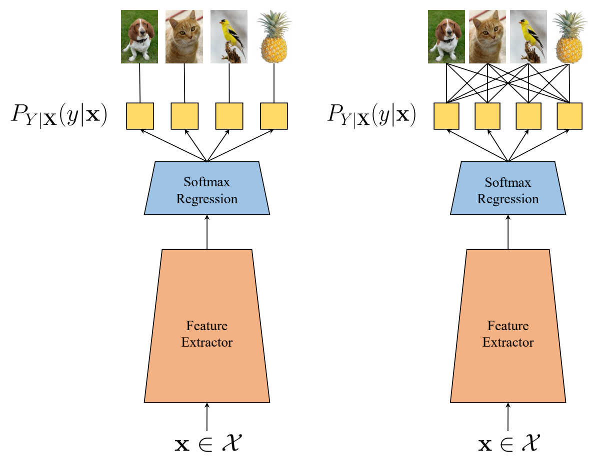



Figure 1. Efficiency of CPC supervision for calibration. Left: Under classic cross-entropy training, each training example only provides significant supervision to the posterior probability of its class label. Right: Under CPC, each training example provides significant supervision to the probabilities of *all* classes.

Various methods have been proposed to calibrate DNN

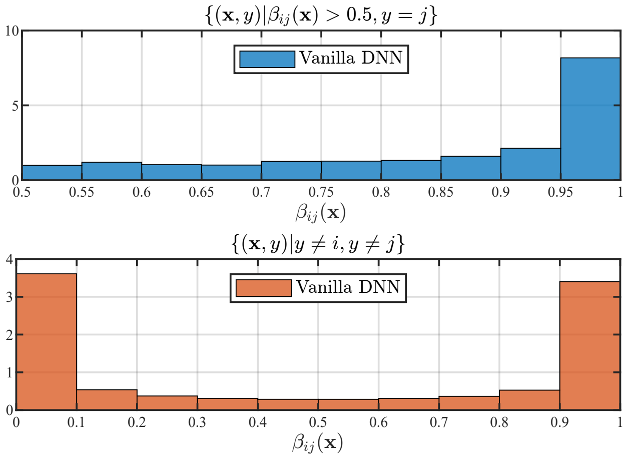



Figure 2. Histograms of the binary posterior probabilities $\beta_{ij}(\mathbf{x})$ produced by a ResNet-101 on CIFAR-100. Top: Examples that belong to the class $j$ and are assigned to the class $i$. Bottom: Examples whose labels are neither class $i$ nor $j$.

probability estimates in the literature, including but not limited to post-processing [6, 38], Bayesian approximation [2, 5], regularization [28, 47], and deep ensemble [22]. These methods have different trade-offs between calibration performance, memory, and computation complexity, with no clear winner when all factors are considered. Their performances also tend to degrade drastically under data shifts [35], i.e. when test examples are corrupted or perturbed [9], a common occurrence for practical applications. Hence, there is a need for robust calibration strategies of low memory footprint and computational complexity.

In this work, we consider this problem, aiming to derive methods that regularize the training of a DNN to encourage better calibration. We address the multiclass classification problem of label set $\mathcal{Y} = \{1, 2, \cdots, C\}$ and hypothesize that poor calibration is due to the inefficient supervision provided by the cross-entropy loss during network training. By establishing as the learning target for each example the one-hot code of the associated class label, this loss encourages myopic training algorithms, which place all emphasis on the posterior probability of the true class and mostly ignore the posterior probabilities of the remaining classes. This is illustrated in Figure 1 where, under classic cross-entropy training, each training example only provides explicit supervision to the posterior probability of the class of the example. While very effective in terms of classification accuracy, this is very inefficient supervision for the purposes of calibration.

To increase the amount of calibration supervision provided per training example, we consider how an example can supervise the classes other than that of its true label. We note that cross-entropy training does this through the constraint that class-posterior probabilities must sum to 1. Hence, a high probability for the true class implies low probabilities for all the alternative classes. This constraint is quite strong for binary classification problems ($C = 2$), where there is only one alternative class, but degrades as $C$ increases, since it is diffused by $C-1$ alternative classes. This suggests the hypothesis that calibration can be strengthened by providing calibration supervision to all class pairs, namely the $C(C-1)/2$ binary classification problems that can be derived from $\mathcal{Y}$. We denote this as *calibration by pairwise constraints* (CPC). In this way, as illustrated in Figure 1, each training example can provide supervision to the posterior probabilities of *all* classes, significantly increasing the degree of supervision over that of cross-entropy training.

In this paper, we start by showing that the proposed CPC has strong theoretical grounding, in that the multiclass posterior probability estimators $\{\pi_y\}_{y\in\mathcal{Y}}$ are calibrated *if and only if* all the derived binary posterior probability estimators $\left\{\beta_{ij} = \frac{\pi_i}{\pi_i+\pi_j}\right\}_{i\neq j\in\mathcal{Y}}$ are calibrated. This provides a simple explanation as to why vanilla DNNs are poorly-calibrated, which is illustrated in Figure 2. The figure shows that the binary posterior estimators $\{\beta_{ij}\}_{ij}$ of a cross-entropy DNN are poorly calibrated in two aspects. First, as shown at the top, when binary estimators that involve the true class $y$ make incorrect predictions, these predictions tend to have high confidence. Second, for binary problems that do not involve the true class, estimators $\beta_{ij}$ ($y \neq i, j$) mostly assign examples to either class $i$ or $j$ with high confidence, instead of producing uncertain predictions.

We then argue that the calibration efficiency of cross-entropy training of a multiclass DNN can be increased by calibrating the binary posterior estimations $\{\beta_{ij}\}_{ij}$, using losses of two types. For class pairs that include the true class $y$, i.e. $\{(i,j)|y\in\{i,j\}\}$, the binary cross-entropy loss is used to encourage $\beta_{ij}$ to assign high probability to class $y$ and low probability to the opposite class. For the remaining pairs $\{(i,j)|y\notin\{i,j\}\}$, an alternative loss is used to encourage $\beta_{ij}$ to give uncertain predictions, outputting the same posterior probability for classes $i, j$.

We finally show that this approach of CPC can be implemented with high computational simplicity. This follows from the fact that the bulk of the computations required by the proposed binary losses are already performed during the standard cross-entropy training of a multiclass network. In fact, we show that the additional losses can be computed by a simple addition of $C(C-1)/2$ sigmoid functions at the top of the network. Hence, CPC allows improved calibration with no increase of memory or time complexity during test and a minor increase in training complexity. Empirical evaluations show that, despite this, CPC calibration achieves state-of-the-art calibration performance across multiple datasets and DNN architectures. The calibration gains of CPC are also shown to increase with the number of classes and example scarcity, i.e. they are larger for smaller training sets. These observations confirm that CPC increases the rate of calibration supervision provided by each example.

Overall, this work makes five contributions. The first is the hypothesis, illustrated in Figures 1 and 2, that the limited supervision provided by the cross-entropy loss for calibration is an important reason for the poor calibration performance of DNNs. The second is the hypothesis that the problem can be addressed through the proposed CPC. The third is theoretical evidence in support of this hypothesis, by showing that the multiclass problem can only be calibrated if all derived binary classifiers are. The fourth is showing that, for DNNs, CPC can be implemented with minimal complexity. Finally, it is shown that training with CPC indeed enables significant improvements in calibration performance, is complementary to existing approaches such as deep ensembles, and enables state-of-the-art calibration performance for several network architectures and datasets.

# 2. Related Works

## 2.1. Probability Calibration of DNNs

Several works have observed that standard training does not produce calibrated DNNs [5, 6, 34]. Various approaches have been proposed to address this problem.

**Post-processing approaches:** The calibration of binary classifiers has been long studied. Methods such as histogram binning [53], isotonic regression [54], Bayesian binning into quantiles [31], and Platt scaling [38] have been proposed prior to the introduction of deep learning. Most of these methods fix the classifier and learn a calibration map by hold-out validation, a posteriori. Most can be extended to the multiclass setting and combined with DNNs. Among them, temperature scaling, the simplest extension of Platt scaling, has been shown the most effective in evaluations [6].

**Regularization:** A few DNN regularization techniques can also improve confidence calibration, although this was not their original goals. Two examples are label smoothing [28] and mixup [47] which are originally proposed to improve generalization [37, 46] and adversarial robustness [55], respectively. In addition, several regularization losses specifically designed for calibration have been proposed [15, 52].

**Bayesian DNNs:** Bayesian neural networks are known for their ability to express uncertainty about their predictions [26, 32]. While exact Bayesian learning and inference are intractable for DNNs, many approximation methods, e.g. Monte Carlo dropout [5] or Bayes by backprop [2], have been proposed [14, 45]. [5] has shown that DNN dropout [44] can be cast as approximate Bayesian inference. [42] generalized this framework to other stochastic inference techniques such as skipping layers [11]. [2] proposed to use stochastic variational inference as an approximate Bayes approach.

**Ensemble:** Deep ensembles [22] average the probability predictions of multiple independently trained DNNs. This was shown to outperform many single-DNN methods discussed above, in terms of both classification and calibration performance [35]. Its major shortcoming is that the memory and time complexity linearly scale with the ensemble size. Several efficient ensemble methods have been proposed [49, 50]. [24] proposed to train a single DNN knowledge distillation [10] from a deep ensemble.

## 2.2. Reducing Multiclass to Binary

In machine learning, a classical approach to multi-class classification is reducing the problem to $C(C-1)/2$ binary problems, since the binary problems are usually much easier to solve [1, 7]. The binary predictions can be combined by simply voting [4] or other pairwise coupling algorithms [40, 51]. This strategy has been successfully employed for multiclass classification using support vector machines [51], AdaBoost [1], and shallow neural networks [39]. However, this strategy has rarely been employed for complicated models like DNNs, partly because its complexity quadratic in $C$ can be prohibitive for DNNs.

# 3. Calibration by Pairwise Constraints

In this section, we first discuss the relationships between the probability calibration of multiclass and pairwise binary classification. We then introduce the approach of CPC.

## 3.1. Multiclass DNNs

A multiclass DNN is a mapping from the feature space $\mathcal{X}$ into a set of labels $\mathcal{Y} = \{1, \ldots, C\}$. The DNN performs classification in three stages. The first is a feature extractor or embedding $\mathbf{v} : \mathcal{X} \rightarrow \mathcal{V} \subset \mathbb{R}^d$ which is parameterized by $\theta$ and maps an observation $\mathbf{x} \in \mathcal{X}$ into a $d$-dimensional feature space $\mathcal{V}$. This is typically achieved through a sequence of layers combining linear and non-linear transformations. The second is estimating the class-posterior probability distribution by a softmax regression

$$\pi_y^\Theta(\mathbf{x}) := P(y|\mathbf{x}; \Theta) = \frac{e^{\langle \mathbf{w}_y, \mathbf{v}(\mathbf{x}) \rangle + b_y}}{\sum_{k=1}^C e^{\langle \mathbf{w}_k, \mathbf{v}(\mathbf{x}) \rangle + b_k}}, \quad (1)$$

where $\mathbf{w}_y/b_y$ is the classification weight/bias for class $y$, $\Theta = \{\theta\} \bigcup \{\mathbf{w}_y, b_y\}_{y=1}^C$, and $\langle \cdot, \cdot \rangle$ denotes the dot product. In what follows, we will omit the dependence of $\pi_y^\Theta(\mathbf{x})$ on $\Theta$ or $\mathbf{x}$, for the sake of simplicity, whenever convenient. The third is the Bayes decision rule

$$y(\mathbf{x}) = \arg\max_i \pi_i^\Theta(\mathbf{x}). \quad (2)$$

A DNN is said to be calibrated if it produces accurate estimates of the class-posterior probability distributions $\pi = (\pi_1, \ldots, \pi_C)$. More precisely, the class-posterior for a given observation $\mathbf{x}$ is said to be calibrated if

$$\pi_i(\mathbf{x}) = \pi_i^*(\mathbf{x}) \quad \forall i, \tag{3}$$

where $\pi_i^*$ is the optimal estimator such that

$$\mathbb{E}_{\mathbf{x},y}\left[\mathbb{1}_{y=i} | \pi_i^*(\mathbf{x}) = p\right] = p, \forall p \in (0, 1], \tag{4}$$

where $\mathbb{1}_.$ is an indicator function that is 1 if its argument is true, and 0 otherwise. The DNN is *perfectly calibrated* if (3) holds for all $\mathbf{x} \in \mathcal{X}$.

### 3.2. Multiclass and Pairwise Binary Calibration

The set of classes $\mathcal{Y}$ also defines many *one-versus-one* (1v1) classification problems. These are binary classification problems opposing class $i$ to class $j$ for all $i \neq j$. Let $\mathcal{B}_{ij}$ be the classification problem opposing class $i$ to class $j$ and $\mathcal{B}(\mathcal{Y}) = \{\mathcal{B}_{ij}\}_{ij}$ be the set of all such problems derived from the set $\mathcal{Y}$. The class-posterior probabilities of the 1v1 problem $\mathcal{B}_{ij}$ are then given by

$$\begin{aligned} \beta_{ij} &= P(y = i | y = i \text{ or } y = j, \mathbf{x}) \\ &= \frac{P(y = i|\mathbf{x})}{P(y = i \text{ or } y = j|\mathbf{x})} \\ &= \frac{\pi_i}{\pi_i + \pi_j} = 1 - \beta_{ji}. \end{aligned} \tag{5}$$

The 1v1 problem is calibrated if

$$\beta_{ij} = \beta_{ij}^* = \frac{\pi_i^*}{\pi_i^* + \pi_j^*}. \tag{6}$$

The following result shows that the binary calibration problems provide alternative constraints for the calibration of the multiclass problem.

**Lemma 1.** *The calibration of all binary problems $\{\mathcal{B}_{ij}(\mathcal{Y})\}_{ij}$ derived from the class set $\mathcal{Y}$ is a necessary and sufficient condition for the calibration of the multiclass problem defined by $\mathcal{Y}$.*

*Proof.* Proof of necessity: Assume that there exists one binary problem $\mathcal{B}_{ij}$ which is not calibrated. Using $\beta_{ji} = 1 - \beta_{ij}$, we have $\beta_{ij} \neq \beta_{ij}^*$ and $\beta_{ji} \neq \beta_{ji}^*$. It follows that

$$\begin{aligned} \beta_{ij} &\neq \beta_{ij}^* \\ \frac{\pi_i}{\pi_i + \pi_j} &\neq \frac{\pi_i^*}{\pi_i^* + \pi_j^*} \\ \pi_i \pi_i^* + \pi_i \pi_j^* &\neq \pi_i^* \pi_i + \pi_i^* \pi_j \\ \pi_i \pi_j^* &\neq \pi_i^* \pi_j \\ \frac{\pi_i}{\pi_j} &\neq \frac{\pi_i^*}{\pi_j^*} \end{aligned}$$

from which it cannot be true that both $\pi_i = \pi_i^*$ and $\pi_j = \pi_j^*$ hold. Hence the multiclass posterior distribution $\pi$ cannot be calibrated. It follows that $\pi$ is calibrated only if all binary problems are calibrated.

Proof of sufficiency: Assume that all binary problems are calibrated. Then $\beta_{ij} = \beta_{ij}^*, \forall i, j$ and it follows that

$$\frac{\beta_{ij}}{\beta_{ji}} = \frac{\beta_{ij}^*}{\beta_{ji}^*} \quad \forall i, j$$

from which

$$\begin{aligned} \frac{\pi_i}{\pi_j} &= \frac{\pi_i^*}{\pi_j^*} \quad \forall i, j \\ \sum_{i \neq j} \frac{\pi_i}{\pi_j} &= \sum_{i \neq j} \frac{\pi_i^*}{\pi_j^*} \quad \forall j \\ \frac{1 - \pi_j}{\pi_j} &= \frac{1 - \pi_j^*}{\pi_j^*} \quad \forall j \\ \pi_j &= \pi_j^* \quad \forall j \end{aligned}$$

and the multiclass problem is calibrated.

□

### 3.3. Supervision Rate for Calibration

Given a training set $\mathcal{D}^{\text{train}} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, the DNN parameters $\Theta$ are learned by minimizing the empirical risk

$$\mathcal{R}(\mathcal{L}) = \sum_{i=1}^{n} \mathcal{L}(\mathbf{x}_i, y_i; \Theta), \tag{7}$$

where $\mathcal{L}$ is a loss function, usually the cross-entropy loss

$$\mathcal{L}^{ce}(\mathbf{x}, y; \Theta) = -\log \pi_y^{\Theta}(\mathbf{x}). \tag{8}$$

We hypothesize that the poor calibration of DNNs trained in this manner is partially due to the fact that cross-entropy training is a very inefficient form of supervision in terms of calibration constraints. Note that (8) only provides explicit supervision to the probability $\pi_y$ of the class $y$ to which $\mathbf{x}$ belongs. While some supervision is implicitly provided to the other classes through the constraint that the posterior probabilities must sum to one, this is very diffuse, not targeting any probability in particular.

Overall, as illustrated in Figure 1, the supervision rate of cross-entropy training for calibration is, roughly speaking, of one class per example, for a total of $\mathcal{O}(n)$ for the entire dataset. Since the dilution of supervision for the posterior probabilities of the classes other than the label $y$ increases with the number of classes $C$, our hypothesis suggests that calibration will degrade as $C$ increases. Experimentally, we have confirmed that the calibration performance of DNNs trained under the cross-entropy loss usually degrades drastically as the number of classes $C$ increases and the number of training examples $n$ decreases. This is discussed in more detail in Section 5 and Figure 4.

## 4. Calibration with Pairwise Constraints

In this section, we consider how to increase the calibration supervision rate of DNN training.

### 4.1. Binary Discrimination Constraints

Since, from Lemma 1, calibration of the multiclass classifier is equivalent to calibration of all binary classification problems derived from $\mathcal{Y}$, we propose to use these problems to increase the supervision rate for calibration of the training process. We start by considering the problems $\mathcal{B}_{ij}$ involving the true class, i.e. $y \in \{i, j\}$. To calibrate these problems, we resort to the binary cross-entropy loss

$$\mathcal{L}_{ij}^{1v1}(\mathbf{x}, y; \Theta) = -\mathbb{1}_{y=i} \log \beta_{ij}(\mathbf{x}) - \mathbb{1}_{y=j} \log \beta_{ji}(\mathbf{x}). \quad (9)$$

Note that, for a given $y$, this is identical to

$$\mathcal{L}_{ij}^{1v1}(\mathbf{x}, y; \Theta) = \begin{cases} -\log \beta_{yj} = -\log \frac{\pi_y}{\pi_y + \pi_j}, & \text{if } y = i, \\ -\log \beta_{yi} = -\log \frac{\pi_y}{\pi_y + \pi_i}, & \text{if } y = j, \\ 0, & \text{otherwise.} \end{cases}$$

The entire pool of binary classifiers can be calibrated by the addition of the 1v1 loss

$$\begin{aligned} \mathcal{L}^{1v1}(\mathbf{x}, y; \Theta) &= \frac{1}{2(C-1)} \sum_{ij} \mathcal{L}_{ij}^{1v1}(\mathbf{x}, y; \Theta) \\ &= \frac{1}{2(C-1)} \left( \sum_{j \neq y} \mathcal{L}_{yj}^{1v1} + \sum_{i \neq y} \mathcal{L}_{iy}^{1v1} \right) \\ &= -\frac{1}{(C-1)} \sum_{j \neq y} \log \frac{\pi_y}{\pi_y + \pi_j} \\ &= -\frac{1}{(C-1)} \sum_{j \neq y} \log \frac{1}{1 + \frac{\pi_j}{\pi_y}}. \end{aligned} \quad (10)$$

This loss provides explicit supervision to the probabilities of all $(C-1)$ class pairs that involve $\pi_y$. We denote these pairs as binary discrimination class pairs, and $\mathcal{L}^{1v1}$ as the *binary discrimination constraints* (BDC) loss. The addition of $\mathcal{L}^{1v1}$ to the cross-entropy loss $\mathcal{L}$ increases the rate of supervision for calibration to $\mathcal{O}(nC)$.

### 4.2. Binary Exclusion Constraints

It remains to consider the binary problems $\{\mathcal{B}_{ij}\}_{ij}$ that do not include the true label $y$, i.e. $y \notin \{i, j\}$. For such problems, the observation does not belong to any of these two classes and the true binary posterior is unknown. In the absence of other information, it is natural to adopt a noninformative prior, i.e. a uniform prior

$$\beta_i^*(\mathbf{x}) = \beta_j^*(\mathbf{x}) = 1/2. \quad (11)$$

Constraint can be included in the training by adding to the previous losses the Kullback-Leibler divergence [20] to this uniform prior distribution

$$\mathcal{L}_{ij}^{be}(\mathbf{x}, y; \Theta) = -\frac{1}{2} \mathbb{1}_{y \neq i, y \neq j} (\log \beta_{ij}(\mathbf{x}) + \log \beta_{ji}(\mathbf{x})). \quad (12)$$

This is denoted as a *binary exclusion constraint* (BEC) because it identifies the two classes as not being responsible for the example $\mathbf{x}$. The BEC loss is then defined as the average of all such constraints,

$$\begin{aligned} \mathcal{L}^{be}(\mathbf{x}, y; \Theta) &= \frac{1}{(C-1)(C-2)} \sum_{ij} \mathcal{L}_{ij}^{be}(\mathbf{x}, y; \Theta) \\ &= -\frac{\sum_{i \neq y, j \neq y} \log \beta_{ij} + \log \beta_{ji}}{2(C-1)(C-2)} \\ &= -\frac{\sum_{i \neq y, j \neq y} \log \frac{\pi_i}{\pi_i + \pi_j} + \log \frac{\pi_j}{\pi_i + \pi_j}}{2(C-1)(C-2)} \\ &= -\frac{\sum_{i \neq y, j \neq y} \log \frac{1}{1 + \frac{\pi_j}{\pi_i}} + \log \frac{1}{1 + \frac{\pi_i}{\pi_j}}}{2(C-1)(C-2)}. \end{aligned} \quad (13)$$

This loss provides explicit supervision to the class-posterior probabilities of all $(C-1)(C-2)$ class pairs that do not include $y$ and increases the rate of supervision for calibration to $\mathcal{O}(nC^2)$.

### 4.3. Implementation

The binary loss functions above are all composed of terms of the form $\frac{1}{1+\frac{\pi_i}{\pi_j}}$. For the softmax classifier of (1),

$$\begin{aligned} \frac{1}{1 + \frac{\pi_i}{\pi_j}} &= \frac{1}{1 + \frac{e^{\langle \mathbf{w}_i, \mathbf{v}(\mathbf{x}) \rangle + b_i}}{e^{\langle \mathbf{w}_j, \mathbf{v}(\mathbf{x}) \rangle + b_j}}} \\ &= \frac{1}{1 + e^{\langle \mathbf{w}_i - \mathbf{w}_j, \mathbf{v}(\mathbf{x}) \rangle + b_i - b_j}} \\ &= \sigma(\langle \mathbf{w}_j - \mathbf{w}_i, \mathbf{v}(\mathbf{x}) \rangle + b_j - b_i) \\ &= \sigma(\langle \mathbf{w}_j, \mathbf{v}(\mathbf{x}) \rangle + b_j - \langle \mathbf{w}_i, \mathbf{v}(\mathbf{x}) \rangle - b_i) \\ &= \sigma(l_j(\mathbf{x}) - l_i(\mathbf{x})), \end{aligned} \quad (14)$$

where $\sigma(u) = (1 + e^{-u})^{-1}$ is the sigmoid function and $l_i(\mathbf{x}) = \langle \mathbf{w}_i, \mathbf{v}(\mathbf{x}) \rangle + b_i$ is the logit computed at the input of the softmax function at the top of the network.

Hence, the loss functions $\mathcal{L}^{1v1}$ and $\mathcal{L}^{be}$ can be written as

$$\mathcal{L}^{1v1}(\mathbf{x}, y; \Theta) = -\frac{1}{C-1} \sum_{j \neq y} \log \sigma(l_y(\mathbf{x}) - l_j(\mathbf{x})), \quad (15)$$

$$\begin{aligned} \mathcal{L}^{be}(\mathbf{x}, y; \Theta) = &- \frac{\sum_{i \neq y, j \neq y} \log \sigma(l_i(\mathbf{x}) - l_j(\mathbf{x}))}{2(C-1)(C-2)} \\ &- \frac{\sum_{i \neq y, j \neq y} \log \sigma(l_j(\mathbf{x}) - l_i(\mathbf{x}))}{2(C-1)(C-2)}. \end{aligned} \quad (16)$$

Finally, the binary calibration constraints can be enforced by combining the two pairwise binary constraints, BDC of

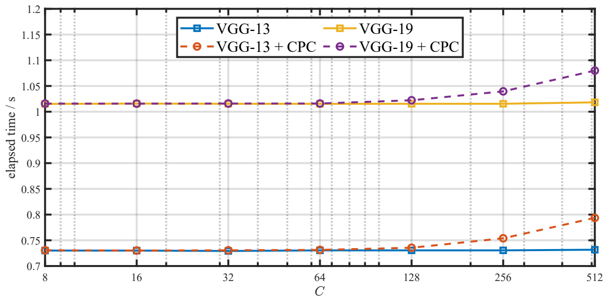


Figure 3. Training time per iteration of processing 256 224×224 images versus $C$ (averaged over 1000 runs on an NVIDIA A40 GPU).

(15) and BEC of (16), with the cross-entropy loss of (8) into an overall objective

$$\mathcal{L} = \lambda_1 \mathcal{L}^{ce} + \lambda_2 \mathcal{L}^{1v1} + \lambda_3 \mathcal{L}^{be}, \tag{17}$$

where $\lambda_1$, $\lambda_2$ and $\lambda_3$ are nonnegative multipliers. Training with this loss is denoted as *Calibration by Pairwise Constraints* (CPC). Note that, because the logits $\{l_i(\mathbf{x})\}_{i=1}^{C}$ are already required for the computation of $\mathcal{L}^{ce}$, the computation of the terms $\{\log \sigma(l_j(\mathbf{x}) - l_i(\mathbf{x}))\}_{ij}$ in (15, 16) has very minimal additional complexity. This is empirically demonstrated in Figure 3, which compares the time cost of training DNNs with and without CPC. For $C \leq 512$, the additional time cost brought by CPC is less than 10% and almost negligible. In summary, CPC enables a significant increase in the rate of supervision for calibration, from $\mathcal{O}(n)$ to $\mathcal{O}(nC^2)$, at the cost of a very minimal increase in training complexity.

# 5. Experiments

## 5.1. Experimental Setup

### 5.1.1 Datasets and Networks

CPC was evaluated on two natural image datasets, CIFAR-10 and CIFAR-100 [16], commonly used in the calibration literature. For evaluation under dataset shift, we used CIFAR-10-C and CIFAR-100-C [9] consisting of images which are first extracted from the test sets of CIFAR-10 and CIFAR-100 and then corrupted by 16 different types of distortions (with 5 levels of intensity each), such as Gaussian blur and JPEG compression. To study the compatibility of CPC with different types of DNNs, evaluations were made with multiple DNN architectures: VGG-13, VGG-19 [43], ResNet-34, and ResNet-101 [8]. A modern technique batch normalization [12] was added for VGG-13 and VGG-19. Since images of CIFAR-10/100 have a low resolution (32×32), we set the stride of the first convolutional layer of ResNet-34 and ResNet-101 to 1.

### 5.1.2 Evaluation Metrics

For any class $i \in \mathcal{Y}$, the corresponding class-posterior probability estimator $\pi_i$ is perfectly calibrated if

$$\mathbb{E}_{\mathbf{x},y}\left[\mathbb{1}_{y=i} | \pi_i(\mathbf{x}) = p\right] - p = 0, \forall p \in (0, 1]. \tag{18}$$

In practice, it is infeasible to verify if (18) holds, since $p$ is a continuous variable and the expectation in LHS of (18) cannot be estimated for all values of $p$ using a finite sample $\mathcal{D}^{\text{test}} = \{(\mathbf{x}_i, y_i)\}_i$. A popular approximate estimation of the calibration error is to quantize the interval $(0, 1]$ into $M$ bins $\{I_m = (\frac{m-1}{M}, \frac{m}{M}]\}_{m=1}^{M}$, define $B_m = \{i | \max_y \pi_y(\mathbf{x}_i) \in I_m\}$ as the index set of the examples assigned to $I_m$, and obtain the accuracy and average confidence of each bin as

$$\text{acc}(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} \mathbb{1}_{y_i = \arg\max_y \pi_y(\mathbf{x}_i)}, \tag{19}$$

$$\text{conf}(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} \max_y \pi_y(\mathbf{x}_i), \tag{20}$$

where $|\cdot|$ denotes the cardinality of a set. Expected calibration error (ECE) [31] and average calibration error (ACE) [33] are then defined as

$$\text{ECE} = \sum_{m=1}^{M} \frac{|B_m|}{|\mathcal{D}^{\text{test}}|} |\text{acc}(B_m) - \text{conf}(B_m)| \tag{21}$$

$$\text{ACE} = \sum_{m=1}^{M} \frac{1}{M} |\text{acc}(B_m) - \text{conf}(B_m)| \tag{22}$$

and employed to evaluate calibration quality in this work.

### 5.1.3 Implementation Details

We implemented CPC using PyTorch [36]. All models were trained by stochastic gradient descent (SGD), with momentum of 0.9 and weight decay of 0.0005, for 200 epochs. SGD batch size was set to be 256. Learning rate was initialized as 0.1 and decayed by 0.2 at epochs 80, 140, 180. For each combination of dataset and network, $\lambda_1$, $\lambda_2$, and $\lambda_3$ in (17) were chosen by a holdout validation on the training set. For the evaluation metrics ECE and ACE, $M$ was set to be 20. See the supplementary material for more implementation details.

## 5.2. Empirical Results

### 5.2.1 Effects of $C$ and $n$ on Calibration

In the discussion above, we hypothesized that CPC increases the rate of supervision for calibration. Roughly speaking, this states that CPC increases the number of calibration constraints per training example. This implies

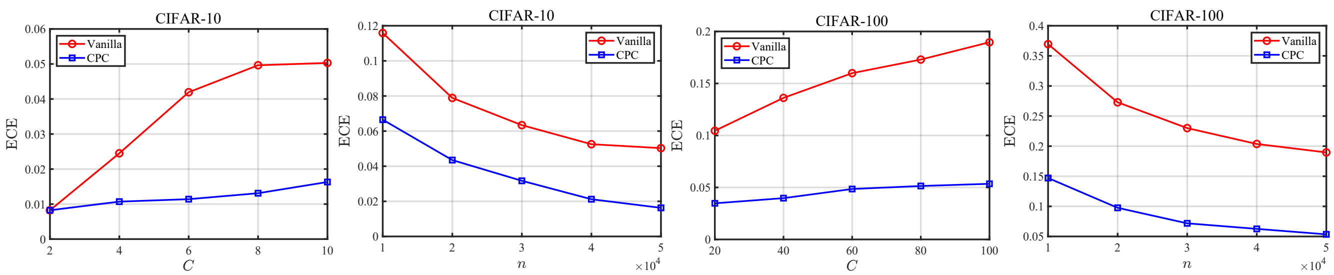



Figure 4. Expected calibration error (ECE) versus the number of classes $C$ and the number of training examples $n$. Evaluations are averaged over 5 runs with a VGG-19 network.

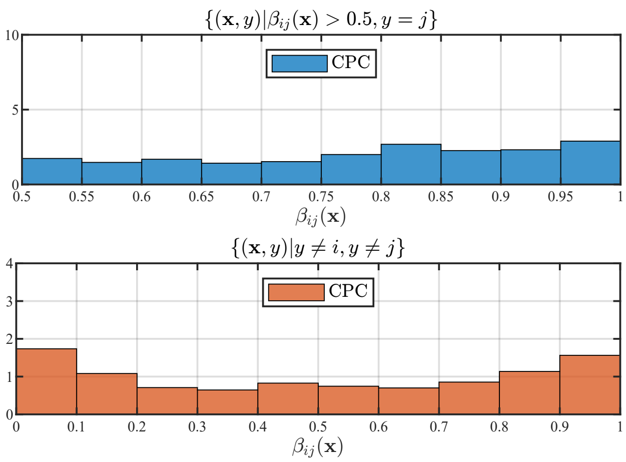



Figure 5. Histograms of the binary posterior probabilities $\beta_{ij}(\mathbf{x})$ produced on CIFAR-100, by a ResNet-101 trained with CPC. Top: Examples that belong to the class $j$ and are assigned to the class $i$. Bottom: Examples whose labels are neither class $i$ nor $j$.

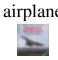
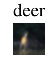
| | airplane  | cat  | deer  | automobile  |
|---|---|---|---|---|
| Vanilla | bird (0.95) | dog (0.64) | horse (0.96) | truck (0.95) |
| CPC | bird (0.71) | dog (0.42) | horse (0.58) | truck (0.81) |

Figure 6. Sample images and their class predictions by different classifiers. Posterior estimations are shown in parenthesis.

that introducing CPC should be equivalent to using vanilla cross-entropy training with a training set of a larger size $n$. In general, it is expected that ECE will be a decreasing function of $n$. The addition of CPC should thus push the curve of ECE vs $n$ to the left. We have also hypothesized that the weak calibration performance of vanilla cross-entropy is due to the fact that each example mostly contributes supervision for the calibration of the true class probability. The remaining probabilities only receive supervision through the constraint that all posterior probabilities must sum to one. Since this constraint is increasingly more diffuse as the number of classes $C$ grows, ECE should increase with $C$ for a given $n$. Because CPC provides supervision to all class-posterior probabilities, its impact should be larger as $C$ increases.

To validate these hypotheses, we evaluated the calibration error of DNNs as a function of $C$ and $n$. This was done by randomly sampling $C$ training classes and $n$ training examples from the original training set. The resulting ECE curves are shown in Figure 4 and confirm both hypotheses. First, the calibration performance of vanilla cross-entropy training degrades drastically with the increase of $C$ and the decrease of $n$. Second, for a fixed number of classes $C$, CPC shifts the curve of ECE vs $n$ to the left. The gains of CPC can be drastic. For example, on CIFAR-100, CPC training with a dataset of 10,000 images achieves better calibration than vanilla training with 50,000 images. Third, for a given dataset of size $n$, CPC shifts the curve of ECE vs $C$ to the right.

#### 5.2.2 Qualitative Results

Figure 5 plots the histograms of the binary probabilities $\beta_{ij}(\mathbf{x})$ of Figure 2, when the ResNet-101 is trained with CPC. The problematic behavior of the vanilla DNN in Figure 2 has been largely alleviated. The network assigns much much lower confidences to its mistakes and more uniform probabilities to the classes other than the true label. This is a typical plot for CPC trained networks across all architectures and datasets considered in this work. A few misclassified images sampled from CIFAR-10 are shown in Figure 6. With CPC, varying degrees of decrease in estimated class-posterior probabilities associated with the incorrect predictions are observed.

#### 5.2.3 Comparison to the State-of-the-art

CPC was compared to several popular single-model calibration baselines: vanilla DNN, temperature scaling [6], MC dropout [44], label smoothing [28, 46], and mixup [47, 55]. For evaluation of MC dropout on ResNet-34 and ResNet-101 which do not use dropout, we inserted a dropout layer

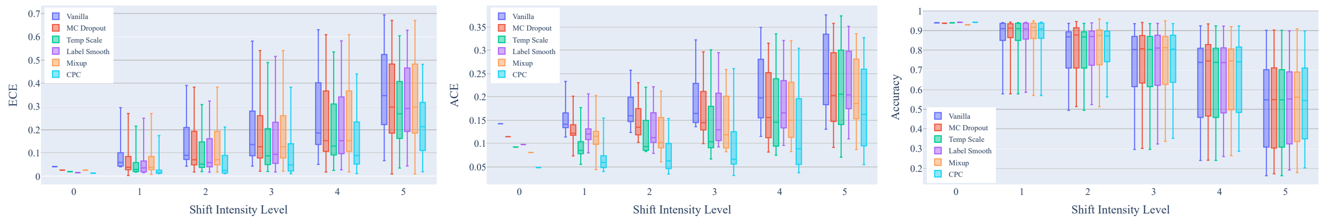



Figure 7. Calibration and classification performance of different methods on VGG-13 and CIFAR-10 under different levels of data shift.

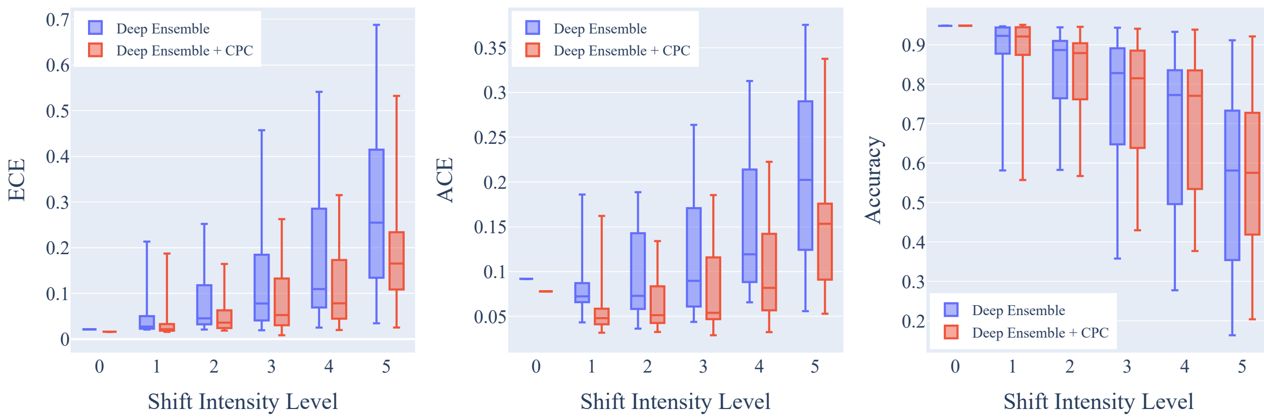



Figure 8. Comparison of the performance of deep ensembles with and without CPC on VGG-19 and CIFAR-10.

between the feature extractor and the classifier.

Figure 7 summarizes the calibration and classification performance of the different methods for VGG-19 on the CIFAR-10 dataset. In this figure, the comparison is limited to single-model approaches, which require a single network during inference. Due to space limitations, the results of other combinations of network architecture and dataset are provided in the supplementary material. Some conclusions can be drawn from the figures. First, CPC always achieves accuracy comparable with the other methods. Second, CPC has the best calibration performance among all single-model methods on both datasets, for almost all network architectures. For many architectures and metrics, the gains can be sizeable.

### 5.2.4 Deep Ensemble with CPC

It is well known that calibration performance can be boosted by using a deep ensemble, i.e. an ensemble of independently trained DNNs. This tends to improve both classification accuracy and calibration performance at the cost of more expensive inference in terms of both memory and computation. CPC is complementary to deep ensembles, since it can be used to calibrate each of the networks in the ensemble. To investigate the benefits of CPC for deep ensembles, we considered ensembles of size 3 and compared an ensemble of vanilla DNNs to an ensemble of DNNs trained with CPC. The results of these experiments are summarized in Figure 8. It is shown that deep ensembles with CPC achieve comparable accuracy and better calibration than vanilla deep ensembles.

## 6. Conclusion

We considered the problem of probability calibration of DNNs. We first showed that the calibration of a $C$-way classifier is equivalent to the calibration of $C(C-1)/2$ pairwise binary classifiers. In light of this, we proposed two pairwise calibration constraints that increase the calibration supervision rate. This was shown to enable state-of-the-art probability calibration performance. In the future, we will investigate the possible limitations of our method, such as whether the complexity of proposed constrains $\mathcal{O}(nC^2)$ will become an issue for super large $C$.

## Acknowledgements

This work was partially funded by NSF awards IIS-1924937 and IIS-2041009, a gift from Amazon, a gift from Qualcomm, and NVIDIA GPU donations. We also acknowledge and thank the use of the Nautilus platform for some of the experiments discussed above.

## References


[1] Erin L Allwein, Robert E Schapire, and Yoram Singer. Reducing multiclass to binary: A unifying approach for margin classifiers. *Journal of machine learning research (JMLR)*, 1(Dec):113–141, 2000. 3

[2] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. In *ICML*, 2015. 2, 3

[3] Glenn W Brier. Verification of forecasts expressed in terms of probability. *Monthly weather review*, 78(1):1–3, 1950. 1

[4] Jerome H. Friedman. Another approach to polychotomous classification. Technical report, Stanford University, Department of Statistics, 1996. 3

[5] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *ICML*, 2016. 1, 2, 3

[6] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *ICML*, 2017. 1, 2, 3, 7

[7] Trevor Hastie and Robert Tibshirani. Classification by pairwise coupling. *The annals of statistics*, 26(2):451–471, 1998. 3

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1, 6

[9] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *ICLR*, 2018. 2, 6

[10] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 3

[11] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *ECCV*, 2016. 3

[12] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 6

[13] Xiaoqian Jiang, Melanie Osl, Jihoon Kim, and Lucila Ohno-Machado. Calibrating predictive model estimates to support personalized medicine. *Journal of the American Medical Informatics Association*, 19(2):263–274, 2012. 1

[14] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? In *NeurIPS*, 2017. 3

[15] Ranganath Krishnan and Omesh Tickoo. Improving model calibration with accuracy versus uncertainty optimization. In *NeurIPS*, 2020. 3

[16] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. 6

[17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, 2012. 1

[18] Volodymyr Kuleshov, Nathan Fenner, and Stefano Ermon. Accurate uncertainties for deep learning using calibrated regression. In *ICML*, 2018. 1

[19] 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 *NeurIPS*, 2019. 1

[20] S. Kullback and R. A. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, Mar. 1951. 5

[21] Ananya Kumar, Percy S Liang, and Tengyu Ma. Verified uncertainty calibration. In *NeurIPS*, 2019. 1

[22] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *NeurIPS*, 2017. 2, 3

[23] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015. 1

[24] Zhizhong Li and Derek Hoiem. Improving confidence estimates for unfamiliar examples. In *CVPR*, 2020. 3

[25] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 1

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

[27] Jishnu Mukhoti, Viveka Kulharia, Amartya Sanyal, Stuart Golodetz, Philip HS Torr, and Puneet K Dokania. Calibrating deep neural networks using focal loss. In *NeurIPS*, 2020. 1

[28] Rafael Müller, Simon Kornblith, and Geoffrey E Hinton. When does label smoothing help? In *NeurIPS*, 2019. 2, 3, 7

[29] Allan H Murphy. A new vector partition of the probability score. *Journal of Applied Meteorology*, 12(4):595–600, 1973. 1

[30] Allan H Murphy and Robert L Winkler. Reliability of subjective probability forecasts of precipitation and temperature. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 26(1):41–47, 1977. 1

[31] Mahdi Pakdaman Naeini, Gregory F Cooper, and Milos Hauskrecht. Obtaining well calibrated probabilities using bayesian binning. In *AAAI*, 2015. 3, 6

[32] Radford M Neal. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012. 3

[33] Lukas Neumann, Andrew Zisserman, and Andrea Vedaldi. Relaxed softmax: Efficient confidence auto-calibration for safe pedestrian detection. In *NeurIPSW*, 2018. 6

[34] Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *CVPR*, 2015. 1, 3

[35] Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, David Sculley, Sebastian Nowozin, Joshua Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. In *NeurIPS*, 2019. 1, 2, 3

[36] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An

imperative style, high-performance deep learning library. In *NeurIPS*, 2019. 6

[37] Gabriel Pereyra, George Tucker, Jan Chorowski, Łukasz Kaiser, and Geoffrey Hinton. Regularizing neural networks by penalizing confident output distributions. In *ICLRW*, 2017. 3

[38] John Platt et al. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74, 1999. 2, 3

[39] David Price, Stefan Knerr, Léon Personnaz, and Gérard Dreyfus. Pairwise neural network classifiers with probabilistic outputs. In *NeurIPS*, 1994. 3

[40] Philippe Refregier and François Vallet. Probabilistic approach for multiclass classification with neural networks. In *ICANN*, 1991. 3

[41] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015. 1

[42] Seonguk Seo, Paul Hongsuck Seo, and Bohyung Han. Learning for single-shot confidence calibration in deep neural networks through stochastic inferences. In *CVPR*, 2019. 3

[43] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 1, 6

[44] 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 (JMLR)*, 15(1):1929–1958, 2014. 3, 7

[45] Mahesh Subedar, Ranganath Krishnan, Paulo Lopez Meyer, Omesh Tickoo, and Jonathan Huang. Uncertainty-aware audiovisual activity recognition using deep bayesian variational inference. In *CVPR*, 2019. 3

[46] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, 2016. 3, 7

[47] 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 *NeurIPS*, 2019. 2, 3, 7

[48] Christian Tomani, Sebastian Gruber, Muhammed Ebrar Erdem, Daniel Cremers, and Florian Buettner. Post-hoc uncertainty calibration for domain drift scenarios. In *CVPR*, 2021. 1

[49] Yeming Wen, Dustin Tran, and Jimmy Ba. Batchensemble: an alternative approach to efficient ensemble and lifelong learning. In *ICLR*, 2020. 3

[50] Florian Wenzel, Jasper Snoek, Dustin Tran, and Rodolphe Jenatton. Hyperparameter ensembles for robustness and uncertainty quantification. In *NeurIPS*, 2020. 3

[51] Ting-Fan Wu, Chih-Jen Lin, and Ruby C Weng. Probability estimates for multi-class classification by pairwise coupling. *Journal of Machine Learning Research (JMLR)*, 5(Aug):975–1005, 2004. 3

[52] Chen Xing, Sercan Arik, Zizhao Zhang, and Tomas Pfister. Distance-based learning from errors for confidence calibration. In *ICLR*, 2020. 1, 3

[53] Bianca Zadrozny and Charles Elkan. Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers. In *ICML*, 2001. 3

[54] Bianca Zadrozny and Charles Elkan. Transforming classifier scores into accurate multiclass probability estimates. In *KDD*, 2002. 3

[55] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *ICLR*, 2018. 3, 7

[56] Zhisheng Zhong, Jiequan Cui, Shu Liu, and Jiaya Jia. Improving calibration for long-tailed recognition. In *CVPR*, 2021. 1