Calibrating a Deep Neural Network with Its Predecessors

Linwei Tao[1] , **Minjing Dong**[1] , **Daochang Liu**[1] , **Changming Sun**[2] and **Chang Xu**[1]
[1]School of Computer Science, Faculty of Engineering, University of Sydney, Australia
[2]CSIRO's Data61, Australia
{linwei.tao, mdon0736, daochang.liu}@sydney.edu.au, changming.sun@csiro.au, c.xu@sydney.edu.au

Abstract

Confidence calibration - the process to calibrate the output probability distribution of neural networks - is essential for safety-critical applications of such networks. Recent works verify the link between mis-calibration and overfitting. However, early stopping, as a well-known technique to mitigate overfitting, fails to calibrate networks. In this work, we study the limitions of early stopping and comprehensively analyze the overfitting problem of a network considering each individual block. We then propose a novel regularization method, predecessor combination search (PCS), to improve calibration by searching a combination of best-fitting block predecessors, where block predecessors are the corresponding network blocks with weight parameters from earlier training stages. PCS achieves the state-of-the-art calibration performance on multiple datasets and architectures. In addition, PCS improves model robustness under dataset distribution shift. Supplementary material and code are available at https://github.com/Linwei94/PCS

1 Introduction

Deep neural networks (DNNs) have achieved great successes across a variety of domains, especially on classification related tasks such as object detection [Wang *et al.*, 2022; Zhou *et al.*, 2021; Qiao *et al.*, 2021] and image classification [Pham *et al.*, 2021; Wortsman *et al.*, 2022], reaching prediction accuracy far beyond human beings. However, they still suffer from mis-calibrated predictions in the sense that the prediction probability cannot represent the ground-truth probability.This may lead to fatal problems when any safety-critical downstream tasks such as autonomous driving [Bojarski *et al.*, 2016] and medical diagnosis [Caruana *et al.*, 2015] rely heavily on the prediction probability.

The underlying cause for mis-calibrated predictions is associated with the capacity of modern neural networks that makes them vulnerable to overfitting [Guo *et al.*, 2017]. [Mukhoti *et al.*, 2020] show that overfitting in modern neural networks mostly results from the overconfidence on misclassified samples and they empirically verify the strong connection between the overfitting issue and calibration performance. Given this observation, some regularization techniques such as weight decay [Guo *et al.*, 2017], label smoothing [Müller *et al.*, 2019], and data augmentation [Thulasidasan *et al.*, 2019; Hendrycks *et al.*, 2019] are introduced to improve model calibration.

Early stopping [Prechelt, 1998] is another well-known regularization method, which suspends training once the model performance stops improving on a hold out validation dataset. [Mukhoti *et al.*, 2020] conduct a series of empirical experiments and demonstrate that early stopping on training according to multiple criteria fails to yield a well-calibrated model. We mainly attribute this sub-optimal solution to the unitary strategy of conventional early stopping techniques which treat the entire network as a whole. Specifically, an early stopping technique takes a DNN as a black box without investigating the internal components, i.e., the blocks inside the network. However, the increasing depth of modern DNNs makes the optimization more challenging, which could lead to discrepancies of convergence speeds of different blocks in DNNs. Thus, any model calibration via a conventional early-stopping technique could be a sub-optimal solution.

In this paper, instead of taking a DNN as a whole, we consider a block in a DNN as the basic unit and explore the overfitting problem in each block. We empirically observe that blocks in a network overfit at different stages during training. Unlike the conventional early stopping approach [Prechelt, 1998] that stops the training of the whole network at a certain point to form a network predecessor, we propose to stop the training of each block at its own best-fitting block predecessor to improve model calibration. However, the blocks in DNNs are strongly coupled with each other, and the early stopping of an individual block independently does not ensure an optimal solution. To achieve an effective and adaptive early stopping for each block, we take into consideration all possible block predecessor combinations. Our objective is to discover the predecessor combination (PC) with better calibration performance. We propose a neural architecture search inspired approach, predecessor combination search (PCS) to calibrate the DNNs, which performs a differential search of the optimal block predecessor combination through a relaxation of the search space as well as a predecessors evaluation estimator.

Our contribution can be summarized as follows: **(1)** We study the overfitting problem of individual blocks empiri-

cally and show that different blocks reach their own overfitting points at different stages of training. **(2)** We propose a novel differential PCS method to search a better-calibrated model together with a sampling strategy to improve searching efficiency. **(3)** PCS achieves state-of-the-art results for both pre- and post-temperature scaling [Guo *et al.*, 2017] on a variety of datasets and architectures via a large number of experiments. We show that PCS works well on out-of-distribution (OoD) samples by shifting the dataset from CIFAR-10 to SVHN [Goodfellow *et al.*, 2013] and CIFAR-10-C [Hendrycks and Dietterich, 2018].

2 Related Works

Due to the mis-calibration problem in modern neural networks [Guo *et al.*, 2017] and the significant importance of calibration, many techniqueshave been proposed in recent years. The current calibration methods could be divided into three categories. The first category modifies the training loss by replacing the conventionally used cross-entropy loss with a mean square error loss [Hui and Belkin, 2020] or a focal loss [Gupta *et al.*, 2020] or by adding an auxiliary regularization loss such as the MMCE loss [Kumar *et al.*, 2018] and the AvUC loss [Krishnan and Tickoo, 2020]. [Bohdal *et al.*, 2021] propose a differentiable surrogate for expected calibration error that improves the calibration performance directly. The recent work in [Karandikar *et al.*, 2021] introduces a differentiable bin membership function and applies it on bin-based metrics such as expected calibration error (ECE) [Guo *et al.*, 2017] to make it become a differentiable auxiliary calibration loss.

Another category is the post-hoc calibration approaches that improve the calibration performance by modifying the prediction logits. Platt scaling [Platt *et al.*, 1999] learns parameters to perform a linear transformation on the original prediction logits. Isotonic regression [Zadrozny and Elkan, 2002] learns piece-wise functions to transform the original prediction logits. Histogram binning [Zadrozny and Elkan, 2001] obtains calibrated probability estimates from decision trees and naive Bayesian classifiers. Bayesian binning into quantiles (BBQ) [Naeini *et al.*, 2015] is an extension of histogram binning with Bayesian model averaging. Beta calibration [Kull *et al.*, 2017] is proposed for binary classification and [Kull *et al.*, 2019] generalize the beta calibration method from binary classification to multi-classification with Dirichlet distributions. [Wenger *et al.*, 2020] employ a non-parametric representation using a latent Gaussian process. Among these methods, temperature scaling is the most popular post-hoc approach, which tunes the temperature parameter of the softmax function that minimizes the negative log likelihood (NLL) and does not change the prediction results. In this work, we present the calibration performance with both before and after temperature scaling.

All other regularization methods that can calibrate networks form the third category. Label smoothing [Müller *et al.*, 2019] implicitly calibrates networks by artificially softening targets to prevent a model from overfitting to the "hard label". Mixup [Thulasidasan *et al.*, 2019] and AugMix [Hendrycks *et al.*, 2019] are two popular data augmentation techniques for calibration. The mix step in data augmentation increases the generality of datasets and reduces the influence of hard samples that can easily cause over-confidence problem. Weight decay, which dominated regularization methods for neural networks in the past, is now less often used by modern neural networks. However, it still plays an important role in improving model calibration [Guo *et al.*, 2017]. Learning with Retrospection(LWR) [Deng and Zhang, 2021] makes use of the learned information in the past epochs to guide the subsequent training, which benefit the classification prediction and uncertainty.

3 Problem Formulation

Considering a dataset $\mathcal{D} = \langle (x_i, y_i) \rangle_{i=1}^{N}$ with N samples from a joint distribution $(\mathcal{X}, \mathcal{Y})$, the ground-truth class label is $y_i \in \{1, 2, ..., \mathcal{K}\}$, where $\mathcal{K}$ denotes the number of classes. The probability for a class y_i on a given input x_i predicted by network F with model parameters Θ is denoted as $\hat{p}_{i,y_i} = F_\Theta(y_i|x_i)$. The predicted label $\hat{y}_i$ and the corresponding confidence $\hat{p}_i$ are defined as

$$\begin{aligned} \hat{y}_i &= \mathrm{argmax}_{y_i \in \{1,2,...,\mathcal{K}\}} \, \hat{p}_{i,y_i}, \\ \hat{p}_i &= \max_{y_i \in \{1,2,...,\mathcal{K}\}} \hat{p}_{i,y_i}. \end{aligned} \tag{1}$$

When the model is *perfectly calibrated*, the prediction confidence $\hat{p}$ is expected to represent the real probability p for each sample x_i with class label y_i. In other words, the model accuracy $\mathbb{P}(\hat{y} = y|\hat{p} = p)$ is p, for all $p \in [0, 1]$.

ECE is a widely-accepted metric to measure calibration performance. Formally, the ECE is defined as the expected absolute difference between the model's confidence and its accuracy, which can be formulated as

$$\mathrm{ECE} = \mathbb{E}_{\hat{p}} \big[\, |\mathbb{P}(\hat{y} = y|\hat{p}) - \hat{p}| \, \big]. \tag{2}$$

Due to the finite samples in datasets, [Guo *et al.*, 2017] estimate ECE by dividing confidence $p \in [0, 1]$ into $\mathbb{B}$ equal-width bins. B_i denotes the set of samples with confidences within $\left(\frac{i-1}{\mathbb{B}}, \frac{i}{\mathbb{B}}\right]$. Let I_i and C_i denote the accuracy and average confidence of all samples in bin B_i respectively. Accuracy of bin B_i is computed as $I_i = \frac{1}{|B_i|} \sum_{j \in B_i} \mathbb{1} \left(\hat{y}_j = y_j\right)$, where $\mathbb{1}$ is the indicator function, and $\hat{y}_j$ and y_j are the predicted and ground-truth labels for the j^{th} sample. Similarly, the confidence C_i of the i^{th} bin is computed as $C_i = \frac{1}{|B_i|} \sum_{j \in B_i} \hat{p}_j$, i.e., C_i is the average confidence of all samples in the bin. Thus, in practice, ECE is formulated as the weighted average of accuracy-confidence difference for the bins:

$$\mathrm{ECE} = \sum_{i=1}^{\mathbb{B}} \frac{|B_i|}{N} \left| I_i - C_i \right|. \tag{3}$$

Along with ECE, the maximum calibration error (MCE) is proposed to minimize the influence of worst-case confidence deviation, which is defined as the maximum difference of bins' accuracy and confidence: $\mathrm{MCE} = \max_{i \in 1,...,\mathbb{B}} |I_i - C_i|$.

[Guo *et al.*, 2017] and [Mukhoti *et al.*, 2020] state that the mis-calibration problem of networks is strongly related to overfitting on training sets. Early stopping, as a well-known

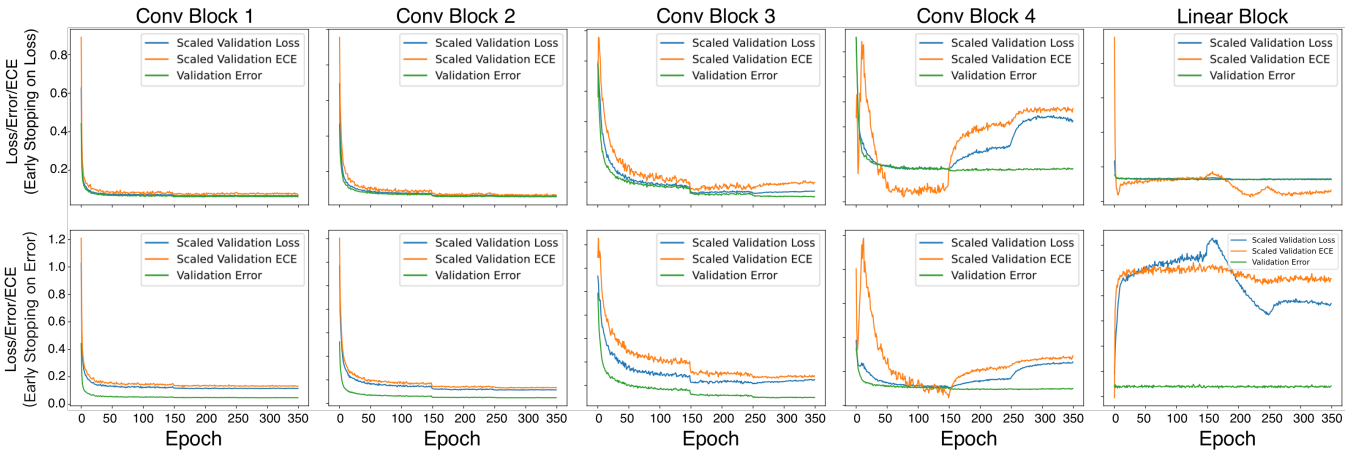

Figure 1: **Empirical evidence on that different blocks have different overfitting behaviors.** A ResNet-50 on CIFAR-10 is first trained with 350 epochs with the cross-entropy loss and learning rate scheduling at epochs 150 and 250. ResNet-50 has 5 blocks, i.e., 4 convolutional blocks and 1 final linear layer. Each sub-figure shows the overfitting issue of block f_i through the training epochs with $\tilde{F}_i$ fixed at potential "sweet point" predecessors. Top row and bottom row represent $\tilde{F}_i$ fixed at different predecessors, which are the "sweet point" epoch of the whole model by early stopping on loss (epoch 151) and error (epoch 281) respectively. Columns represent different block f_i under study. All combinations are fine-tuned with one epoch on the training set.

regularization technique, mitigates the overfitting problem by stopping the whole network at the best-fitting epoch. According to previous observations, early stopping should yield a well calibrated model. However, the calibration performance of an early stopped model is often far from satisfactory. Thus, we hypothesize that shallow and deep blocks may suffer from different degrees of overfitting during optimization and a unified regularization criteria to the entire network F_Θ could lead to a sub-optimal solution for model calibration. Instead, it would be more desirable to resolve the overfitting problems block-wise and explore an adaptive regularization approach to improve the model calibration performance.

3.1 Overfitting in Blocks

Most advanced CNN model architectures are the stack of blocks. Suppose there are M blocks in the network, the model parameters can be represented as $\Theta = \{\theta_i | i = 1, 2, \ldots, M\}$, where θ_i denotes the parameter of the i^{th} block. Θ^j and θ^j are used to identify the network and block parameters at the j^{th} training epoch. The degree of fitting of F_Θ is represented by the trend of validation NLL loss $\mathcal{L}_{NLL}(F_{\Theta^j}(x), y), j = 1, 2, ..., T_{train}$, where T_{train} is the number of total training epochs over the training process. Conventional early stopping aims to find the weights at a "sweet point"[1] epoch to balance between underfitting and overfitting. However, there is no definition of overfitting of individual network blocks. In order to find the "sweet point" epoch of each block, we need to measure the degree of overfitting of an individual block f_i. We formally define the ***predecessors*** of block f_i as $\{f_i^j | j \in \{1, 2, \ldots, T_{train}\}\}$, where f_i^j denotes the i^{th} block with weight at the j^{th} epoch. To achieve the overfitting measurement of f_i, we treat the other blocks of the network $\tilde{F}_i = \{f_{i^-} | i^- \in \{1, 2, \ldots, M\} \setminus \{i\}\}$ as constant mappings. Then the overfitting degree of block f_i could be represented by the trend of validation NLL loss over the training process $\mathcal{L}_{NLL}(F(x), y; f_i^j), j = 1, 2, ..., T_{train}$, with the weights of other network blocks $\tilde{F}_i$ fixed. Based on this idea, an empirical study is conducted to investigate the overfitting behaviors of individual network blocks.

We present the empirical study in Figure 1, demonstrating the different overfitting behaviors of individual network blocks. A ResNet-50 is first trained on CIFAR-10 for $T_{train} = 350$ epochs. Then each sub-figure shows the evaluation of overfitting behavior of block f_i and each block other than f_i is fixed to a certain predecessor. Note that it is non-trivial to properly select the fixed predecessors for all other blocks $\tilde{F}_i$ and ideally these blocks need to be at their own "sweet points". We adopt a simplified approach and fix $\tilde{F}_i$ at the "sweet point" of the whole model as an approximation, i.e., $\tilde{F}_i = \{f_{i^-}^{\rho^*} | i^- \in \{1, 2, \ldots, M\} \setminus \{i\}\}$, where ρ^* is obtained either by early stopping the whole model based on the validation loss or classification error. Apart from the validation loss, the validation ECE and classification error are also plotted in Figure 1. The proportionally scaled values of the validation loss and ECE are reported for better visualisation. Through the variation of the validation loss of each block f_i in this experiment as shown in Figure 1, we have the following intriguing observations on overfitting in blocks.

1. Deeper convolutional blocks tend to have more severe overfitting problems with training. From the first four columns in Figure 1, we observe that all convolutional blocks with the same block index share a similar overfitting pattern for both settings. However, when we look into each row, a different pattern shows on the convolutional blocks compared with each other. The deeper convolutional block, i.e., f_4,

[1] A sweet point is defined as the balance point between underfitting and overfitting, which normally refers to the lowest point of the validation loss curve.

starts rapid overfitting after the first learning rate scheduler point.

2. The overfitting problem of the final linear layer is much more complex and depends heavily on other blocks. When it comes to the final linear block f_5 at the last column, the loss presents a totally reverse pattern with different $\tilde{F}_i$. The final block f_5 with $\tilde{F}_i$ fixed at predecessors early stopping on loss has a slight overfitting problem from epoch 50 to the first learning rate scheduler point at epoch 150, while f_5 with $\tilde{F}_i$ fixed at predecessors early stopping on classification error continues to overfit from the very first epoch until a few epochs after the first learning rate scheduler point while the error remains at the same level.

3. Mis-calibration is linked with block overfitting. When we look at the variation of ECE and validation loss, we observe that the overfitting trend of an individual block is consistent with the variation of ECE, which further extends the link between mis-calibration and overfitting at block-wise level.

According to these observations, it could be very difficult to identify the ideal PC due to the inter-dependency among blocks (the linear blocks in the two rows have totally different behaviours). In other words, simply finding the best-fitting blocks individually and combining them together without taking other blocks into account may not produce a best-fitting model. This motivates us to explore an automatic searching algorithm to find a group of predecessors at "sweet points" that allows $F^* = \{f_i^{\rho_i^*} | i = 1, 2, \ldots, M\}$ to best-fit to the training set. Here, ρ_i^* denotes the optimal block predecessor choice of block f_i.

4 Methodology

Based on the observation on Figure 1, we propose to explore a PC representation $\mathcal{P} = \{\rho_i | i = 1, 2, \ldots, M\}$ to tackle the overfitting issue in blocks and improve model calibration performance. The objective can be formulated as

$$\min_{\mathcal{P}} \left(ERR(F) + \lambda \cdot ECE(F) \right) \tag{4}$$

where F is $\{f_i^{\rho_i} | i = 1, 2, \ldots, M\}$, λ denotes a hyperparameter, $\rho_i \in \{1, 2, \ldots, T_{train}\}$ indicates the predecessor selection of block f_i, ERR and ECE are the classification error and ECE respectively. The direct optimization of Eq. (4) is not feasible via gradient descent due to two issues. First, the discrete representation $\mathcal{P}$ makes Eq. (4) an optimization problem over a discrete domain. Second, both terms ERR and ECE in our objective are not differentiable. Thus, we introduce a PCS framework to tackle the aforementioned issues. Specifically, we first introduce differentiable combination sampling through a continuous relaxation of the PC representation. Proxy classification error and ECE landscape are then introduced to achieve differential optimization of our objective through an estimator for predicting the classification error and ECE.

4.1 Differentiable Combination Sampling

In our PCS framework, the objective is to discover the optimal selection of predecessors from candidate sets for each block. To learn the selection, we first exploit a K-dimensional trainable parameter $\alpha_i \in \mathbb{R}^K$ for this predecessor selection, where K denotes the number of candidates. And ρ_i can be obtained by the argmax of the selection parameter α_i and further represented in a one-hot encoding format $\rho_i \in \mathbb{R}^K$. The PC representation $\mathcal{P}$ can be written as:

$$\begin{aligned} &\mathcal{P} = \{\rho_i | i = 1, 2, \ldots, M\}, \\ \text{s.t.} \quad &\rho_i = \text{one-hot}(\operatorname{argmax} \alpha_i). \end{aligned} \tag{5}$$

To relax the discrete PC representation for gradient-based optimization, we use the Gumbel-Softmax trick [Jang *et al.*, 2016] to approximate the one-hot distribution and introduce randomness. The ρ_i in Eq. (5) can be relaxed as:

$$\tilde{\rho}_i^k = \frac{\exp((\alpha_i^k + \xi_i^k)/\tau)}{\sum_{k'=1}^{K} \exp((\alpha_i^{k'} + \xi_i^{k'})/\tau)}, \tag{6}$$

where τ is the temperature parameter, ξ_i^k is an i.i.d sample from Gumbel(0, 1), k and k' denote the k^{th} and k'^{th} logit of corresponding K-dimensional vector respectively. We denote the relaxed PC representation as $\tilde{\mathcal{P}} = \{\tilde{\rho}_i | i = 1, 2, \ldots, M\}$. With the learning of α_i, $\tilde{\mathcal{P}}$ explores the random combination at the beginning of search and gradually converges to a relatively stable state.

For simplicity, we denote the model with $\tilde{\mathcal{P}}$ as $F_{\tilde{\mathcal{P}}}$. The classification error and ECE on the validation set can be denoted as $ERR(\mathcal{D}_{val}, F_{\tilde{\mathcal{P}}})$ and $ECE(\mathcal{D}_{val}, F_{\tilde{\mathcal{P}}})$ respectively. Since the combination is hard-combined, we fine-tune $F_{\tilde{\mathcal{P}}}$ with one more epoch on $\mathcal{D}_{train}$, denoted as $F_{\tilde{\mathcal{P}}}^*$. In PCS, the original objective (Eq. (4)) can be reformulated as:

$$\min_{A} \left(ERR(\mathcal{D}_{val}, F_{\tilde{\mathcal{P}}}^*) + \lambda \cdot ECE(\mathcal{D}_{val}, F_{\tilde{\mathcal{P}}}^*) \right), \tag{7}$$

where $A = \{\alpha_i | i = 1, 2, \ldots, M\}$ is the collection of learnable selection parameters for all blocks.

4.2 Proxy Classification Error and ECE Landscape

For differential optimization of PC representation $\tilde{\mathcal{P}}$, we use a trainable estimator ψ to obtain proxies $E\hat{R}R, E\hat{C}E = \psi(\tilde{\mathcal{P}})$ to approximate the classification error and ECE. Since the input $\tilde{\mathcal{P}}$ is a sequential data, we utilize a one-layer long short-term memory (LSTM) to build the estimator ψ: $\mathbb{R}^{M \times K} \to \mathbb{R}^d$ mapping $\tilde{\mathcal{P}}$ to a d-dimensional embedding vector and a linear layer: $\mathbb{R}^d \to \mathbb{R}^2$ outputting $E\hat{R}R$ and $E\hat{C}E$. The estimator ψ is trained with a weighted mean squared error loss function:

$$\begin{aligned} \min_{\psi} L(\psi) = &\frac{1}{T_{se}} \sum_{t=1}^{T_{se}} (E\hat{R}R^{(t)} - ERR^{(t)}(\mathcal{D}_{val}, F_{\tilde{\mathcal{P}}}^*))^2 \\ &+ \gamma (E\hat{C}E^{(t)} - ECE^{(t)}(\mathcal{D}_{val}, F_{\tilde{\mathcal{P}}}^*))^2, \end{aligned} \tag{8}$$

where γ is a hyperparameter to control the loss ratio of ECE, T_{se} is the total searching steps and the superscript (t) indicates the evaluation results at the t^{th} time step. All pairs of $\tilde{\mathcal{P}}$ and its corresponding classification error and ECE are

Dataset	Model	Weight Decay [Guo *et al.*, 2017]		Brier Loss [Brier *et al.*, 1950]		MMCE [Kumar *et al.*, 2018]		Label Smoothing [Szegedy *et al.*, 2016]		FL-3 [Mukhoti *et al.*, 2020]		FLSD-53 [Mukhoti *et al.*, 2020]		PCS Ours	
		Pre T	Post T	Pre T	Post T	Pre T	Post T	Pre T	Post T	Pre T	Post T	Pre T	Post T	Pre T	Post T
CIFAR-100	ResNet-50	17.52	3.42(2.1)	6.52	3.64(1.1)	15.32	2.38(1.8)	7.81	4.01(1.1)	5.13	1.97(1.1)	4.5	2.0(1.1)	**2.0**	**2.0(1.0)**
	ResNet-110	19.05	4.43(2.3)	7.88	4.65(1.2)	19.14	3.86(2.3)	11.02	5.89(1.1)	8.64	3.95(1.2)	8.56	4.12(1.2)	**1.76**	**1.76(1.0)**
	Wide-ResNet-26-10	15.33	2.88(2.2)	4.31	2.7(1.1)	13.17	4.37(1.9)	4.84	4.84(1.0)	2.13	2.13(1.0)	3.03	1.64(1.1)	**1.92**	**1.55(1.1)**
	DenseNet-121	20.98	4.27(2.3)	5.17	2.29(1.1)	19.13	3.06(2.1)	12.89	7.52(1.2)	4.15	1.25(1.1)	3.73	1.31(1.1)	**2.75**	**1.18(1.1)**
CIFAR-10	ResNet-50	4.35	1.35(2.5)	1.82	1.08(1.1)	4.56	1.19(2.6)	2.96	1.67(0.9)	1.48	1.42(1.1)	1.55	0.95(1.1)	**0.80**	**0.53(1.1)**
	ResNet-110	4.41	1.09(2.8)	2.56	1.25(1.2)	5.08	1.42(2.8)	2.09	2.09(1.0)	1.55	1.02(1.1)	1.87	1.07(1.1)	**0.57**	**0.57(1.0)**
	Wide-ResNet-26-10	3.23	0.92(2.2)	1.25	1.25(1.0)	3.29	0.86(2.2)	4.26	1.84(0.8)	1.69	0.97(0.9)	1.56	0.84(0.9)	**0.99**	**0.43(1.2)**
	DenseNet-121	4.52	1.31(2.4)	1.53	1.53(1.0)	5.1	1.61(2.5)	1.88	1.82(0.9)	1.32	1.26(0.9)	1.22	1.22(1.0)	**0.78**	**0.78(1.0)**
Tiny-ImageNet	ResNet-50	15.32	5.48(1.4)	4.44	4.13(0.9)	13.01	5.55(1.3)	15.23	6.51(0.7)	1.87	1.87(1.0)	1.76	1.76(1.0)	**1.32**	**1.32(1.0)**

Table 1: **Calibration Performance.** ECE (%), being the lower the better, is evaluated for different methods. Both pre and post temperature scaling (Pre T and Post T in Table) results are reported. The optimal temperature is obtained on the validation set and is included in brackets.

stored in a memory Π to optimize estimator ψ. After each searching step t, memory Π is updated by $\Pi = \Pi \cup \{(\tilde{\mathcal{P}}^{(t)} : (ECE^{(t)}, ERR^{(t)}))\}$. We can then use the optimized estimator ψ^* to reformulate PCS objective Eq. (7):

$$\begin{aligned} &\min_{A}(E\hat{R}R^{*} + \lambda E\hat{C}E^{*}), \\ &\text{where } E\hat{R}R^{*}, E\hat{C}E^{*} = \psi^*(\tilde{\mathcal{P}}). \end{aligned} \tag{9}$$

The gradients of $E\hat{R}R^{*}$ and $E\hat{C}E^{*}$ can be used to optimize $\tilde{\mathcal{P}}$ and thus A:

$$A' \leftarrow A - \eta \cdot \nabla_A(E\hat{R}R^{*} + \lambda E\hat{C}E^{*}), \tag{10}$$

where A' is the new predecessor selection parameter and η is the learning rate. At the next searching time step, the corresponding $\tilde{\mathcal{P}}'$ is based on A', and memory Π is updated to $\Pi = \Pi \cup \{(\tilde{\mathcal{P}}' : (ECE^{'}, ERR^{'})\}$.

Remark: Search Procedure We first train model F_Θ with T_{train} epochs and randomly store weight parameters at K different epochs. After that, we randomly initialize a warm-up population $\mathbb{H}$ of size $\mathcal{S}$ and evaluate the classification error $ERR^{(t)}(\mathcal{D}_{val}, F^*_{\tilde{\mathcal{P}}_i})$ and ECE $ECE^{(t)}(\mathcal{D}_{val}, F^*_{\tilde{\mathcal{P}}_i})$ of $\tilde{\mathcal{P}}_i \in \mathbb{H}$. The combination-performance pairs are then stored to memory Π, which is used to warm up estimator ψ and equip ψ with prior knowledge about classification error and ECE before searching. After the initial training of model and warming up of ψ, the searching procedure is conducted with T_{se} steps. In each step, one PC representation $\tilde{\mathcal{P}}^{(t)}$ is sampled based on current $A^{(t)}$. The corresponding $F_{\tilde{\mathcal{P}}^{(t)}}$ is then fine-tuned with one epoch and evaluated to obtain a validation error and ECE for the training of ψ. $A^{(t)}$ is then optimized with the proxy classification error $E\hat{R}R$ and proxy ECE $E\hat{C}E$.

Remark: Search Space Sampling Strategy Each selection ρ is an integer between 1 and training epoch T_{train}, and thus the candidate size K is T_{train}. However, storing weight parameters over all training epochs can be very storage-intensive. For instance, storing all candidates of a ResNet-50 training with 350 epochs can take up to 33GB. Considering that the model parameters of adjacent epochs are similar, especially at late training stages, we propose four sampling strategies to reduce the size of search space and improve storage efficiency, which are (1) *Random Sampling*, randomly sampling K different epochs ranging from 1 to T_{train}; (2) *Uniform Sampling*, uniformly sampling K epochs ranging from 1 to T_{train}; (3) *Laplace Sampling*, due to model parameters changing much faster at earlier epochs, sampling K epochs ranging from 1 to T_{train} with a Laplace distribution centering at 0; (4) *Piece-Wise Laplace Sampling*, due to model parameters changing much faster at earlier epochs of each learning schedule, sampling K epochs ranging from 1 to T_{train} with multiple Laplace distributions centering at 0 and other learning schedule epochs (150 and 250 in our case). Since the searching difficulty grows exponentially with the size of K, another benefit of small candidate size is that it improves the searching efficiency.

5 Experiments

5.1 Experimental Settings

Datasets We conduct experiments on various datasets, including CIFAR-10/100 [Krizhevsky, 2012] and Tiny-ImageNet [Deng *et al.*, 2009] to evaluate the calibration performance. We also include the robustness evaluation on Out-of-Distribution (OoD) datasets, including SVHN [Goodfellow *et al.*, 2013] and CIFAR-10-C [Hendrycks and Dietterich, 2018].

Baselines To verify the effectiveness of our proposed algorithm, we include different networks for evaluation, including ResNet-50, ResNet-110 [He *et al.*, 2016], Wide-ResNet-26-10 [Zagoruyko and Komodakis, 2016] and DenseNet-121 [Huang *et al.*, 2017], and compare with various approaches, including training with weight decay at 5×10^{-4} (we find that weight decay at 5×10^{-4} performs the best among multiple values), Brier Loss [Brier *et al.*, 1950], MMCE loss [Kumar *et al.*, 2018], Label smoothing [Szegedy *et al.*, 2016] with a smoothing factor $\alpha_{LS} = 0.05$, focal loss [Mukhoti *et al.*, 2020] with regularisation parameter $\gamma_{focal} = 3$, and scheduled focal loss FLSD-53 [Mukhoti *et al.*, 2020] which uses $\gamma_{focal} = 5$ for $\hat{p} \in [0, 0.2)$ and $\gamma_{focal} = 3$ for $\hat{p} \in [0.2, 1)$.

Other Calibration Metrics Recent works [Nixon *et al.*, 2019; Kumar *et al.*, 2019; Roelofs *et al.*, 2022; Gupta *et al.*, 2020] point out the defects of ECE. To evaluate our method comprehensively, we evaluate our method on three additional calibration metrics, i.e., MCE, Adaptive-ECE [Ding *et al.*, 2020] and classwise-ECE [Kull *et al.*, 2019] along with ECE. We also measure PCS with reliability plots in supplementary.

Training Setup For training on CIFAR-10/100, we set $T_{train} = 350$. The learning rate is set to 0.1 for epoch 0 to 150, 0.01 for 150 to 250, and 0.001 for 250 until the end

Dataset	Model	Weight Decay [Guo *et al.*, 2017]		Brier Loss [Brier *et al.*, 1950]		MMCE [Kumar *et al.*, 2018]		Label Smoothing [Szegedy *et al.*, 2016]		FL-3 [Mukhoti *et al.*, 2020]		FLSD-53 [Mukhoti *et al.*, 2020]		PCS Ours	
		Pre T	Post T	Pre T	Post T	Pre T	Post T	Pre T	Post T	Pre T	Post T	Pre T	Post T	Pre T	Post T
CIFAR-10/SVHN	ResNet-50	94.32	94.56	93.59	93.72	85.17	64.75	78.88	78.89	88.28	88.42	92.48	92.79	**98.12**	**97.04**
	ResNet-110	61.71	59.66	94.80	95.13	85.31	85.39	68.68	68.68	96.74	**96.92**	90.83	90.97	**96.77**	96.77
	Wide-ResNet-26-10	96.82	97.62	94.51	94.51	97.35	97.95	84.63	84.66	98.19	98.05	**98.29**	**98.20**	97.55	97.84
	DenseNet-121	84.43	81.57	94.65	94.66	85.88	84.87	78.79	78.94	89.48	89.42	89.59	89.59	**96.72**	**96.72**
CIFAR-10/CIFAR-10-C	ResNet-50	86.23	86.03	**90.21**	**90.13**	89.97	90.11	72.01	72.02	89.44	89.56	89.45	89.56	89.73	89.79
	ResNet-110	77.53	75.16	84.09	83.86	71.96	70.02	72.17	72.18	82.27	82.18	85.05	84.70	**88.1**	**88.27**
	Wide-ResNet-26-10	81.06	80.68	85.03	85.03	82.17	81.72	71.10	71.16	82.17	81.86	87.05	87.30	**89.62**	**89.95**
	DenseNet-121	87.61	86.41	87.38	87.38	84.9	84.88	73.67	73.8	87.12	87.53	89.47	89.47	**89.52**	**89.52**

Table 2: **Robustness on Dataset Shift**. AUROC (%), being the higher the better, is evaluated for different methods with models shifting from CIFAR-10 (in-distribution) to SVHN and CIFAR-10-C as the OoD datasets.

of training. For training on Tiny-ImageNet, we set $T_{train} = 100$. We follow the same training and validation set spilt setting as [Mukhoti *et al.*, 2020]. The learning rate is set to 0.1 for epoch 0 to 40, 0.01 for epoch 40 to 60, and 0.001 for 60 until the end of training. The fine-tuning learning rate is set to 10^{-4} for CIFAR-10, 5×10^{-4} for CIFAR-100, and 10^{-3} for Tiny-ImageNet. The searching process is performed with $T_{se} = 100$ steps. The population size is $\mathcal{S} = 100$. Experiments are conducted with ResNet-50 on CIFAR-10 if there is no other specification. All networks are optimized using the SGD optimizer with a weight decay at 5×10^{-4} and a momentum of 0.9. The training batch size is set to 128. All experiments are conducted on a single Tesla V-100 GPU with all random seeds set to 1. Our code and results of comparison method are based on the public code and the pre-trained weight provided by [Mukhoti *et al.*, 2020].

Temperature Scaling Following the setting in the prior work [Mukhoti *et al.*, 2020], the temperature parameter τ is optimized by grid searching with $\tau \in [0, 0.1, 0.2, \ldots, 10]$ on the validation set and finding the one with the best post-temperature-scaling ECE, which is also applied on the additional calibration metrics.

5.2 Calibration Performance

We report ECE(%) (computed using 15 bins) along with optimal temperatures in Table 1. PCS achieves the state-of-the-art ECE across all models and datasets and outperforms previous works by large margins, especially pre-temperature-scaling results. More specifically, most PCS pre-temperature-scaling results have already substantially exceeded the post-temperature-scaling results of previous works. The result of ResNet-110 on CIFAR-100 achieves the best calibration performance compared to previous works, with a 7% decrease in ECE. For comparison approaches, the model trained with focal loss is broadly better-calibrated than other methods. However, it fails in some cases such as the evaluation on Tiny-ImageNet. In addition, the scheduled γ_{focal} trick does not always work better than fixed γ_{focal} and it is hard to ascertain which is the better between FL-3 and FL53. The MMCE auxiliary loss performs worst before temperature scaling. Another notable point of PCS is that multiple results such as those with ResNet-110 on CIFAR-10/100 achieve the innately calibrated model (T=1.0), which means that the PC itself has already yielded a well-calibrated model without temperature scaling.

We also evaluate PCS on other widely-accepted metrics including Adaptive ECE, Classwise-ECE and test set error. PCS also achieves the state-of-the-art calibration results on almost all cases. The test set error on Tiny-ImageNet shows a 8.52% decrease from 49.81% to 41.29%, which is mainly because of PCS performing as an early stopping trick.

5.3 Robustness on Out-of-Distribution(OoD) Datasets

A well-calibrated model helps improve the model robustness on OoD datasets [Thulasidasan *et al.*, 2019]. However, temperature scaling is known to be fragile under dataset distribution shift [Ovadia *et al.*, 2019]. PCS form innately calibrated models and thus perform well on OoD datasets. We utilize AUROC (the higher the better) to evaluate the robustness under dataset shift. Table 2 shows the AUROC (%) computed for models trained on CIFAR-10 and tested on the OoD datasets SVHN and CIFAR-10-C. Our method achieves competitive results on almost all cases. The results after temperature scaling tend to drop generally, and approaches yielding better pre-temperature-scaling ECE have better robustness on OoD datasets. Although focal loss works well on calibration, it fails under dataset shift. Our PCS with ResNet-50 on CIFAR-10 achieves a 4% increase compared to previous methods.

5.4 Searching Results

We visualize the searching results of ResNet-50 on CIFAR-10 in Figure 2. The searching results are obtained based on well-fitting (test set loss < 2) PCs. The result shows that the last convolutional block (Conv Block 4) tends to choose predecessors in the first half of training (epoch 50 to 180) while Conv Block 2 and Conv Block 3 prefer predecessors in the second half (epoch 150 to 350). This observation might indicate that the later Conv Blocks tend to overfit earlier than the former ones. The first Conv Block and the final linear block show no particular preference to certain predecessors.

This result is consistent with the evaluation of overfitting of individual blocks in Figure 1. The Conv Block 1 suffers from little overfitting throughout training and thus has no preference to certain predecessors, while the middle blocks (Conv Block 2, 3, 4) prefer predecessors with a low validation loss as shown in Figure 1. We visualize the searching results of other models and observe the similar pattern. We also test our algorithm on other networks such as ViT [Dosovitskiy *et al.*, 2020] and MLP-mixer [Tolstikhin *et al.*, 2021], which show the similar calibration effect.

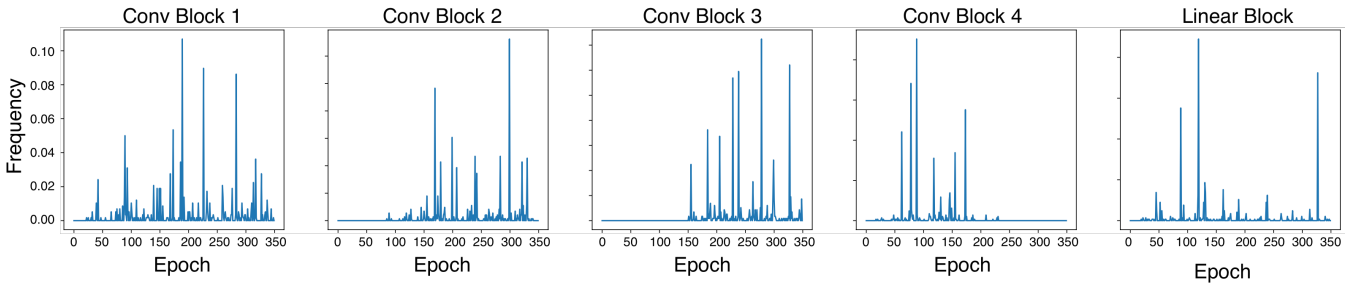

Figure 2: **Searching Results**. The sub-figures show the predecessor choice frequency of different blocks in ResNet-50. The search results are filtered by cross-entropy loss (being less than 0.2) on the test set.

Methods	ERR	ECE	ECE(T)
Early Stopping (Loss)	6.58	2.32	0.56(1.4)
Early Stopping (Error)	4.89	4.03	1.42(2.4)
Early Stopping (ECE)	16.92	1.71	1.09(1.2)
Random Search	5.04	1.13	0.84(1.1)
Search on Loss	5.43	1.1	0.37(1.10)
PCS (λ=10)	5.31	0.86	0.73(1.10)
PCS (λ=25)	5.1	0.75	0.37(1.10)
PCS (λ=50)	5.25	0.58	0.58(1.0)
PCS (λ=100)	5.29	0.70	0.44(1.10)

Table 3: **Comparison of Searching Methods**. Random search is conducted 5 times with ResNet-50 on CIFAR-10. All searching results are selected by the lowest testing loss.

5.5 Ablation Study

Comparison with Other Searching Methods In Table 3, our method is compared with different searching methods as well as early stopping methods. When early stopping on the model as a whole, it is hard to ensure a low error and good calibration performance at the same time. Early stopping on loss and ECE shows a large performance drop on classification error. The random search is conducted 5 times to make the results stable and achieve a relatively high performance on classification error but not ECE. We also compare multiple objectives in Eq. (9) with the performance of a single objective optimization on validation NLL loss, i.e.,$\min_A \hat{NLL}$. Searching on ECE or error individually could lead to extremely unbalanced results. PCS achieves a better result on both test set error and ECE. The hyperparameter λ is tuned on models and datasets.

Weight Sampling Strategy To compare sampling strategies discussed in the previous section, a scatter plot in Figure 3 shows the searching results. Note that the "Full Sampling" indicates searching on all possible predecessors without sampling, i.e., $K = T_{train}$. We use the same $K = 50$ for all sampling strategies. From Figure 3, we observe little difference between different sampling strategies, which indicates that we save storage space with little loss of performance. Thus, we use the random sampling strategy throughout the paper due to its simplicity.

Warm-up Population A larger population indicates more searching time. To find a balance between searching time and performance, we compare the searching results with different population sizes. We use the number of well-fitting results (test loss under 0.2) to measure the searching performance.

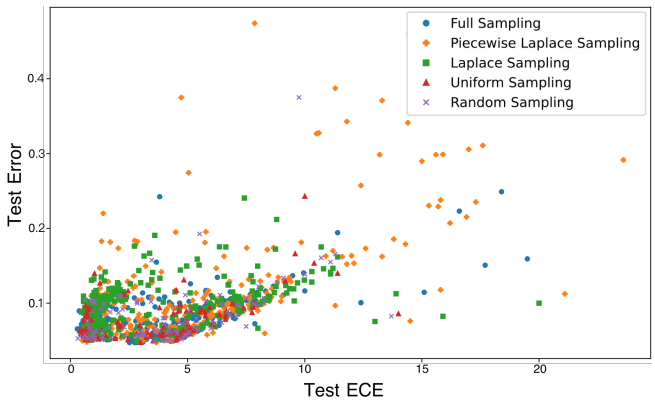

Figure 3: **Comparison of Sampling Strategy**. All experiments are conduct 5 runs with ResNet-50 on CIFAR-10. Searching step T_{se} is set to 100 and produces 500 searching results for each strategy. Metrics along the x-axis and y-axis are the lower the better.

All experimental results are averaged over 5 runs. According to Table 4, the larger the population size, the more well-fitting results can be found since estimator ψ can have better prior knowledge on error and ECE landscape. However, we use a smaller population size as long as the searching provides satisfactory results.

Population Size	64	100	200	500
GPU Hours	3.1	3.7	5.5	10.5
Well-fitting Results	14.8	16	17.8	25.8

Table 4: **Warm-up Population**. Well-fitting results indicate the number of searching results that achieve a testing loss under 0.2, being the higher the better. All results are reported as an average of 5 runs with ResNet-50 on CIFAR-10.

6 Conclusion

In this paper, we address a common problem, the miscalibration in modern neural networks. We observe that different blocks in a network have different overfitting patterns. Our proposed predecessor combination search, as a regularization method, is very effective for calibrating models and can also be potentially applied to other tasks such as learning with noisy labels and improving model robustness.

Acknowledgments

This work was supported in part by the Australian Research Council under Project DP210101859 and the University of Sydney Research Accelerator (SOAR) Prize. The authors acknowledge the use of the National Computational Infrastructure (NCI) which is supported by the Australian Government, and accessed through the NCI Adapter Scheme and Sydney Informatics Hub HPC Allocation Scheme.

References

[Bohdal *et al.*, 2021] Ondrej Bohdal, Yongxin Yang, and Timothy Hospedales. Meta-calibration: Meta-learning of model calibration using differentiable expected calibration error. *arXiv preprint arXiv:2106.09613*, 2021.

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

[Brier *et al.*, 1950] Glenn W Brier, Simon Kornblith, and Geoffrey E Hinton. Verification of forecasts expressed in terms of probability. *Monthly weather review*, 78(1):1–3, 1950.

[Caruana *et al.*, 2015] Rich Caruana, Yin Lou, Johannes Gehrke, Paul Koch, Marc Sturm, and Noemie Elhadad. Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1721–1730, 2015.

[Deng and Zhang, 2021] Xiang Deng and Zhongfei Zhang. Learning with retrospection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 7201–7209, 2021.

[Deng *et al.*, 2009] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[Ding *et al.*, 2020] Yukun Ding, Jinglan Liu, Jinjun Xiong, and Yiyu Shi. Revisiting the evaluation of uncertainty estimation and its application to explore model complexity-uncertainty trade-off. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 4–5, 2020.

[Dosovitskiy *et al.*, 2020] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[Goodfellow *et al.*, 2013] Ian J Goodfellow, Yaroslav Bulatov, Julian Ibarz, Sacha Arnoud, and Vinay Shet. Multi-digit number recognition from street view imagery using deep convolutional neural networks. *arXiv preprint arXiv:1312.6082*, 2013.

[Guo *et al.*, 2017] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International conference on machine learning*, pages 1321–1330. PMLR, 2017.

[Gupta *et al.*, 2020] Kartik Gupta, Amir Rahimi, Thalaiyasingam Ajanthan, Thomas Mensink, Cristian Sminchisescu, and Richard Hartley. Calibration of neural networks using splines. *arXiv preprint arXiv:2006.12800*, 2020.

[He *et al.*, 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[Hendrycks and Dietterich, 2018] Dan Hendrycks and Thomas G Dietterich. Benchmarking neural network robustness to common corruptions and surface variations. *arXiv preprint arXiv:1807.01697*, 2018.

[Hendrycks *et al.*, 2019] 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. *arXiv preprint arXiv:1912.02781*, 2019.

[Huang *et al.*, 2017] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[Hui and Belkin, 2020] Like Hui and Mikhail Belkin. Evaluation of neural architectures trained with square loss vs cross-entropy in classification tasks. *arXiv preprint arXiv:2006.07322*, 2020.

[Jang *et al.*, 2016] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

[Karandikar *et al.*, 2021] Archit Karandikar, Nicholas Cain, Dustin Tran, Balaji Lakshminarayanan, Jonathon Shlens, Michael C Mozer, and Becca Roelofs. Soft calibration objectives for neural networks. *Advances in Neural Information Processing Systems*, 34:29768–29779, 2021.

[Krishnan and Tickoo, 2020] Ranganath Krishnan and Omesh Tickoo. Improving model calibration with accuracy versus uncertainty optimization. *Advances in Neural Information Processing Systems*, 33:18237–18248, 2020.

[Krizhevsky, 2012] Alex Krizhevsky. Learning multiple layers of features from tiny images. *University of Toronto*, 05 2012.

[Kull *et al.*, 2017] Meelis Kull, Telmo Silva Filho, and Peter Flach. Beta calibration: a well-founded and easily implemented improvement on logistic calibration for binary classifiers. In *Artificial Intelligence and Statistics*, pages 623–631. PMLR, 2017.

[Kull *et al.*, 2019] 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. *Advances in neural information processing systems*, 32, 2019.

[Kumar *et al.*, 2018] Aviral Kumar, Sunita Sarawagi, and Ujjwal Jain. Trainable calibration measures for neural networks from kernel mean embeddings. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2805–2814. PMLR, 10–15 Jul 2018.

[Kumar *et al.*, 2019] Ananya Kumar, Percy S Liang, and Tengyu Ma. Verified uncertainty calibration. *Advances in Neural Information Processing Systems*, 32, 2019.

[Mukhoti *et al.*, 2020] Jishnu Mukhoti, Viveka Kulharia, Amartya Sanyal, Stuart Golodetz, Philip Torr, and Puneet Dokania. Calibrating deep neural networks using focal loss. *Advances in Neural Information Processing Systems*, 33:15288–15299, 2020.

[Müller *et al.*, 2019] Rafael Müller, Simon Kornblith, and Geoffrey E Hinton. When does label smoothing help? *Advances in neural information processing systems*, 32, 2019.

[Naeini *et al.*, 2015] Mahdi Pakdaman Naeini, Gregory Cooper, and Milos Hauskrecht. Obtaining well calibrated probabilities using bayesian binning. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[Nixon *et al.*, 2019] Jeremy Nixon, Michael W Dusenberry, Linchuan Zhang, Ghassen Jerfel, and Dustin Tran. Measuring calibration in deep learning. In *CVPR Workshops*, volume 2, 2019.

[Ovadia *et al.*, 2019] 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. *Advances in neural information processing systems*, 32, 2019.

[Pham *et al.*, 2021] Hieu Pham, Zihang Dai, Qizhe Xie, and Quoc V Le. Meta pseudo labels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11557–11568, 2021.

[Platt *et al.*, 1999] John Platt, Nicholas Cain, Dustin Tran, Balaji Lakshminarayanan, Jonathon Shlens, Michael C Mozer, and Becca" Roelofs. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74, 1999.

[Prechelt, 1998] Lutz Prechelt. Early stopping-but when? In *Neural Networks: Tricks of the trade*, pages 55–69. Springer, 1998.

[Qiao *et al.*, 2021] Siyuan Qiao, Liang-Chieh Chen, and Alan Yuille. Detectors: Detecting objects with recursive feature pyramid and switchable atrous convolution. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10213–10224, 2021.

[Roelofs *et al.*, 2022] Rebecca Roelofs, Nicholas Cain, Jonathon Shlens, and Michael C Mozer. Mitigating bias in calibration error estimation. In *International Conference on Artificial Intelligence and Statistics*, pages 4036–4054. PMLR, 2022.

[Szegedy *et al.*, 2016] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[Thulasidasan *et al.*, 2019] Sunil Thulasidasan, Gopinath Chennupati, Jeff A Bilmes, Tanmoy Bhattacharya, and Sarah Michalak. On mixup training: Improved calibration and predictive uncertainty for deep neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

[Tolstikhin *et al.*, 2021] Ilya O Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, et al. Mlp-mixer: An all-mlp architecture for vision. *Advances in neural information processing systems*, 34:24261–24272, 2021.

[Wang *et al.*, 2022] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. *arXiv preprint arXiv:2207.02696*, 2022.

[Wenger *et al.*, 2020] Jonathan Wenger, Hedvig Kjellström, and Rudolph Triebel. Non-parametric calibration for classification. In *International Conference on Artificial Intelligence and Statistics*, pages 178–190. PMLR, 2020.

[Wortsman *et al.*, 2022] Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International Conference on Machine Learning*, pages 23965–23998. PMLR, 2022.

[Zadrozny and Elkan, 2001] Bianca Zadrozny and Charles Elkan. Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers. In *Icml*, volume 1, pages 609–616. Citeseer, 2001.

[Zadrozny and Elkan, 2002] 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*, pages 694–699, 2002.

[Zagoruyko and Komodakis, 2016] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

[Zhou *et al.*, 2021] Xingyi Zhou, Vladlen Koltun, and Philipp Krähenbühl. Probabilistic two-stage detection. *arXiv preprint arXiv:2103.07461*, 2021.