Better Features, Better Calibration: A Simple Fix for Overconfident Networks

Soumya Suvra Ghosal[1] Ramya Hebbalaguppe[2] Dinesh Manocha[1]
[1]University of Maryland [2] TCS Research Labs

Abstract. A model is considered perfectly calibrated when the predicted probabilities align accurately with the true likelihood of the associated classes being correct. Past studies have shown that Deep Neural Networks (DNNs) are susceptible to overfitting and generate miscalibrated predictions. In this paper, we identify that the miscalibration problem in DNNs can be traced back to the features learned by the network. To this end, we propose a new training approach called `RelCal`, which guides the model to focus on a subset of relevant features for each class. Our empirical analysis highlights that training with `RelCal` helps mitigate overconfidence in DNNs, leading to better-calibrated models in terms of Expected Calibration Error (ECE) and Adaptive Expected Calibration Error (AECE). We demonstrate the state-of-the-art results on a diverse range of 8 image classification datasets across architectures spanning CNNs to Transformer-based architectures in terms of network calibration without compromising discriminative performance. Compared to the current best calibration technique, RankMixup [32], `RelCal` reduces the ECE by 4.25% on the challenging imbalanced dataset CIFAR-100-LT. Additionally, on the large-scale ImageNet dataset, `RelCal` reduces the AECE from 9.45% to 3.08%— a 6.37% improvement over the baseline model trained with NLL loss.

Keywords: Confidence Calibration · Reliability · Uncertainty Quantification

1 Introduction

DNNs excel in tasks like object detection, classification, and segmentation but often produce miscalibrated, overconfident predictions. However, despite achieving impressive classification accuracy, they often produce miscalibrated and overconfident predictions [11, 28, 29]. Analysis by [11] revealed that due to large model capacity, neural networks typically become prone to overfitting the negative log-likelihood (NLL) loss during training, thereby essentially prioritizing higher accuracy at the expense of well-calibrated predictions.

The primary issue with miscalibrated predictions lies in providing a false sense of correctness, i.e., the prediction probability associated with a given class may overestimate the likelihood of the class being correct. This issue carries significant implications, especially in safety-critical domains like autonomous driving [10] and healthcare applications [8], where the model must not only be accurate but also correctly confident [11].

There have been numerous efforts to tackle the problem. [11] proposed Temperature Scaling, i.e., dividing the logit outputs by a scalar temperature constant $T > 0$ before performing the softmax operation. Other than temperature scaling, various post-hoc techniques [30, 35, 43] have been introduced to improve calibration during inference. To mitigate model overfitting and miscalibration, [29] suggested training with label-smoothing, and [28] proposed training with focal loss [26]. Recently, [15] argued that in contrast to post-hoc calibration methods that rely on a small set of parameters, train-time strategies leveraging the extensive array of learnable parameters within a DNN offer stronger calibration performance. While many studies propose new training losses for better calibration, we trace miscalibration to the features learned by the model and refine them during training for improved calibration.

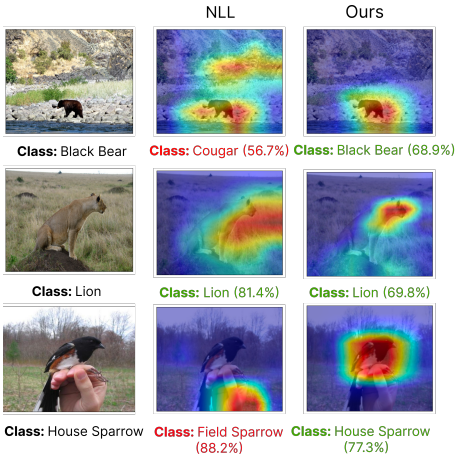


Fig. 1: We illustrate via Grad-CAM visualizations for `ResNet-50` feature extractor trained on `ImageNet` with and without using `RelCal`. Notice that training with `RelCal` allows the network to focus on relevant features of the category enabling better discrimination.

When processing an input image, DNNs typically learn a feature representation which is then passed through a dense linear layer to generate the logit output. Intuitively, each dimension within these feature embeddings can be associated to represent some semantic attribute present in the image [17, 24]. Using a dense layer means that the prediction for a particular class is influenced by contributions from all feature dimensions. Our approach is based on the idea that not all feature dimensions carry equal significance for a particular class. Our approach prioritizes class-relevant features—for example, whiskers and fur matter for "Cat" class but not for "Snake" class.

Main Contributions. Building on this rationale, we propose `RelCal`, a simple and novel approach centered on training a model to prioritize a subset of features that are crucial for each class. To pinpoint critical features for a specific class, we introduce a **relevance score**, computed as the absolute value of the feature dimension's contribution to that class. Our analysis (see Section 4) suggests that `RelCal` introduces an implicit regularization on the model weights, thereby reducing the weight norm. Additionally, we noted that training with `RelCal` demonstrated an increase in NLL loss on accurately classified test samples (see Fig. 2), indicating decreased output confidence. Therefore, based on our observations, we infer that `RelCal` enhances model calibration by imposing regularization on the network weights, consequently mitigating overconfident pre-

dictions. These findings align with the analysis in [28] which identified increase in weight norm as a contributing factor to miscalibration. Further, our observations also highlight that training with `RelCal` not only enhances calibration but also improves test accuracy.

1. **A simple and novel approach, `RelCal` for improving model calibration**. During training, `RelCal` aims to alleviate overfitting by guiding the model to focus on a subset of relevant features that are crucial for each class. Training with `RelCal` leads to reduction in norm of model weights and prediction confidence thereby improving model calibration.
2. **We highlight the strong calibration performance of `RelCal`:** (a) on 5 standard image classification benchmarks (SVHN, CIFAR-10, CIFAR-100, TinyImageNet-200, and ImageNet-1k), (b) two imbalanced datasets (CIFAR-10-LT, and CIFAR-100-LT), (c) one fine-grained image classification dataset (CUB-200), and (d) one NLP dataset (20 Newsgroups). For example, on CUB-200 [42], a fine-grained image classification dataset, using ResNet-101 architecture, our approach reduces the Expected Calibration Error from 8.41% to 2.99% — a **5.42**% of improvement compared to focal loss [28].
3. **`RelCal` is architecture agnostic.** We conduct extensive experimentation using models of varying capacities from ResNet [14] (see Table 6), WideResNet [44] (see Table 7) and Vision Transformer (see Appendix) architectures. Our findings demonstrate that `RelCal` significantly improves model calibration regardless of the architectural choice.

2 Related Works

Calibration Techniques. In deep neural networks (DNNs), the issue of miscalibration was first brought to light by [11]. Their analysis unveiled that DNN models trained with NLL loss tend to exhibit unwarranted overconfidence, leading to a discrepancy between output confidence and actual accuracy. The severity of this issue has sparked considerable research into mitigating this problem. Broadly, approaches to calibrate DNNs can be categorized into two main strategies: train-time calibration and posthoc calibration.

Train-time Calibration approaches introduce additional loss terms/regularization during the training process to enhance model calibration. [34] proposed adding additional entropy loss; [21] proposed MMCE, an auxiliary loss term computed using a reproducing kernel in a Hilbert space; [29] used Label Smoothing (LS) on soft targets; [28] showed that using Focal loss (FL) [26] can be beneficial in preventing the model from becoming overconfident. [15] proposed training with an additional MDCA loss to explicitly minimize the difference in confidence and accuracy for all the classes. Other representative works are MbLS, MixUp, Confidence Ranking Calibration, ACLS [4, 27–29, 33, 45]. Our approach on the contrary utilizes a relevance score for confidence calibration.

Post-hoc Calibration methods operate on models after they have been trained, typically utilizing a validation set. A widely-used post-hoc calibration technique is Temperature Scaling [11], which involves the division of logit outputs by a

scalar temperature constant $T > 0$ before applying the softmax operation. Apart from temperature scaling, several other post-hoc approaches [30,35,43] have been proposed that transform the model output during inference to improve calibration. Dirichlet calibration [19] builds on Dirichlet distributions and extends the Beta-calibration [20] approach, which was originally designed for binary classification, to a multi-class setting. Posthoc calibration assumes the availability of a validation dataset to tune the hyperparameters on the test set and may not generalize well incase of distribution drift.

Pruning Approaches. Extensive research explores training-time pruning to enhance DNN sparsity [2, 12, 13, 39]. The closest methods to our proposed approach include ReAct [40], DICE [41], LINe [1], and ASH [6] which explore *test-time* sparsification mechanism for OOD detection, lacking explicit learning of crucial features. `RelCal` differs from these studies as it serves as a *training-time* regularization technique focusing on learning relevant features to enhance model calibration.

3 Background

Notations. In this paper, we consider a supervised multi-class classification problem. Formally, we consider a training set $\mathcal{D}_{\text{train}}$ consisting of N training samples: $\{\mathbf{x}_i, y_i\}_{i=1}^{N}$. The samples are drawn *i.i.d.* from a probability distribution: $\mathcal{P}_{\mathcal{X},\mathcal{Y}}$. Here, $\mathbf{x} \in \mathcal{X}$ is a random variable defined in the image space, and $y \in \mathcal{Y} = \{1, \ldots, K\}$ represents its label. Traditionally, a parameterized model $f_\theta : \mathcal{X} \to \mathcal{Y}$ is trained on samples drawn from the marginal distribution $\mathcal{P}_{\text{in}}$ of $\mathcal{X}$. The standard aim is to minimize the expected loss $\mathbb{E}_{(\mathbf{x},y)\sim\mathcal{P}}[l(f_\theta(\mathbf{x}), y)]$ under the training distribution $\mathcal{P}$, for some loss function $l : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$.

Calibration. Consider a sample $\mathbf{x} \in \mathcal{X}$ input to a classifier f_θ parameterized by θ. The logit output of the classifier is represented as $\mathbf{f}(\mathbf{x}, \theta) \in \mathbb{R}^K$. The confidence probabilities $\mathbf{p} \in \mathbb{R}^K$ are typically obtained by applying a softmax operation to the logit vector $\mathbf{f}(\mathbf{x}, \theta)$. Let $\hat{y} = \underset{k\in\mathcal{Y}}{\arg\max}\ \mathbf{f}(\mathbf{x}, \theta)[k]$ represent the class predicted by f. Correspondingly, the confidence for the predicted class is computed as $\hat{p} = \mathbf{p}[\hat{y}]$. A model is defined as perfectly calibrated [11] when:

$$\mathbb{P}\left(\hat{y}_i = y_i \mid \hat{p}_i = p_i\right) = p_i, \quad \forall p_i \in (0, 1) \tag{1}$$

Intuitively, if a perfectly calibrated classifier predicts an output with confidence $\hat{p} = 0.8$, then the prediction is accurate 80% of time. The model is over-confident if the prediction accuracy is less than 80% and is under-confident if the prediction accuracy is more than 80%.

4 Our Approach: `RelCal`

Our proposed training approach is designed to enhance the model's focus on relevant features crucial for each class. To identify these important feature dimensions, we first define the notion of *relevance* (see Section 4.1). Next, given an input image, we select a subset of features by thresholding the calculated

feature-wise relevance score. In what follows, we provide an in-depth overview of our proposed approach RelCal.

4.1 Learning the Relevant Features

For feature extraction, we consider a deep neural network f_θ parameterized by θ. Given an input $\mathbf{x} \in \mathcal{X}$, let $\mathbf{h}(\mathbf{x}) \in \mathbb{R}^L$ represent the features extracted from the penultimate layer of the model. The final output $\mathbf{f}(\mathbf{x})$ is obtained by passing $\mathbf{h}(\mathbf{x})$ through a dense linear layer with weight matrix $\mathbf{W} \in \mathbb{R}^{L \times K}$:

$$\mathbf{f}(\mathbf{x}, \theta) = \mathbf{W}^{\mathrm{T}}\mathbf{h}(\mathbf{x}) + \mathbf{b} \tag{2}$$

where $\mathbf{b} \in \mathbb{R}^K$ is the bias vector. Typically, the penultimate layer embeddings $\mathbf{h}(\mathbf{x})$ serve as a proxy for the features learned by a neural net [38]. Hence, we calculate the feature-wise relevance score based on penultimate embeddings $\mathbf{h}(\mathbf{x})$.

Relevance Score. Our main idea is to define a notion of relevance for each feature dimension in $\mathbf{h}(\mathbf{x}) = [h^{(1)}, h^{(2)}, ..., h^{(L)}]$. This is primarily motivated by the observation that the classification of a specific example is dependent on a subset of pertinent attributes. Further, the significance of a particular feature dimension can fluctuate across various classes and instances. Revisiting the "Cat" vs "Snake" example, attributes like whiskers and fur hold significance for a "Cat" image, but may not be relevant for an image from the "Snake" class. To formulate the notion of significance between a feature dimension $h^{(l)}$, where $l \in \{1, \cdots, L\}$, and a class $c \in \{1, \cdots, K\}$, an intuitive relevance score can be defined as:

$$r(h^{(l)}, c) = |h^{(l)}\mathbf{W}_{l,c}|, \tag{3}$$

where $\mathbf{W}_{l,c}$ represents the weight connecting the feature dimension $h^{(l)}$ & the c-th class, and $|\cdot|$ is the absolute value operator. The term $h^{(l)}\mathbf{W}_{l,c}$ essentially quantifies the contribution of the feature $h^{(l)}$ to the c-th class. Finally, based on Eqn. 3, we construct a relevance matrix $\mathbf{R} \in \mathbb{R}^{L \times K}$ such that $\mathbf{R}[i, j] = r(h^{(i)}, j)\ \forall\ i \in \{1, \cdots, L\}, j \in \{1, \cdots, K\}$.

Mask Matrix. Given a feature dimension $h^{(l)}$, its contribution for each class can be measured using the relevance score $r(h^{(l)}, \cdot)$. Based on these scores, we can identify the classes for which $h^{(l)}$ is significant by thresholding on the calculated relevance scores. Specifically, for each feature dimension $h^{(l)}$ we generate a mask vector $\mathbf{m}^{(l)} \in \{0, 1\}^K$ in which each element is 0/1 and is defined as:

$$\mathbf{m}^{(l)}[j] = \begin{cases} 1 & \text{if } r(h^{(l)}, j) \geq \delta^{(l)} \\ 0 & \text{if } r(h^{(l)}, j) < \delta^{(l)} \end{cases} \ \forall j \in \{1, \cdots, K\}, \tag{4}$$

where, $\delta^{(l)}$ represents the threshold constant for the feature dimension $h^{(l)}$. To modulate the threshold, we define a pruning percentile p. Specifically, the threshold $\delta^{(l)}$ is set as the p-th percentile of the scores in the vector $\mathbf{R}[l, :]$. Next, we define the mask matrix $\mathbf{M} \in \mathbb{R}^{L \times K}$ as: $\mathbf{M} = [\mathbf{m}^{(1)}; \mathbf{m}^{(2)}; \cdots; \mathbf{m}^{(L)}]$.

4.2 Training

During training, given an input sample $\mathbf{x}$, we use the mask matrix to prune out any insignificant connection between a feature dimension and class. Specifically, after pruning, the logit output for the c-th class is given by:

$$\mathbf{f}_{\text{rel}}(\mathbf{x}, \theta)[c] = \sum_{l=1}^{L} (\mathbf{R} \odot \mathbf{M})^{\top}[c, l] + \mathbf{b}[c] \ , \tag{5}$$

where, $\odot$ indicates the Hadamard product. Note that, our formulation also flexibly allows each feature dimension to be used for multiple class predictions. As an example, a class "Laptop" might depend on the two most relevant feature dimensions: keyboard and screen. Similarly, the class prediction for "TV" might rely on two features telecontrol and screen. In both cases, the feature of screen is responsible for the classes "TV" and "Laptop". Finally, our training objective takes the form:

$$\min_{\theta} \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{P}_{\mathcal{X}\mathcal{Y}}} \left[\mathcal{L}\left(\mathbf{f}_{\text{rel}}(\mathbf{x};\theta), y\right)\right], \tag{6}$$

where $\mathcal{L}$ is any standard classification loss.

In Appendix, we provide a detailed algorithmic description and PyTorch-based pseudocode of our proposed method. Further, in Section 4.3, we show that training `RelCal` does not lead to any additional computational overhead. The complete code will be made available following paper's acceptance.

Understanding why `RelCal` improves calibration. Existing studies [11, 28] have shown that a significant factor contributing to miscalibration lies in the tendency to overfit the training loss, leading to overconfident yet incorrect predictions. Furthermore, [28] have also pinpointed that the increase in logit magnitudes can be linked to an increase in the norm of the model weights $\|\mathbf{W}\|$. To understand, how `RelCal` improves model calibration, in Fig. 2 we visualize the evolution of different metrics during training the model.

1. **`RelCal` reduces overconfident predictions.** In Fig. 2(a), we compare the test NLL loss for the correctly classified samples throughout training for models trained with and without `RelCal`. We observe that the loss for the `RelCal` trained model remains consistently higher, indicating relatively lower confidence in correctly predicted samples. This observation is further corroborated by Fig. 2(d), where we track the confidence of correctly predicted test samples. It is noteworthy that although `RelCal` reduces confidence in predictions, there is an enhancement in test accuracy (see Tab. 3). Additionally, Fig. 2(b) illustrates that the `RelCal`-trained model demonstrates a lower NLL loss for the misclassified samples. This indicates that for incorrect predictions, `RelCal` (w. NLL) generates relatively higher confidence for the correct class compared to the NLL-trained model, consequently resulting in: (a) reduced test NLL and, (b) decreased confidence for the misclassified class. This observation is also further verified in Fig. 2(e).

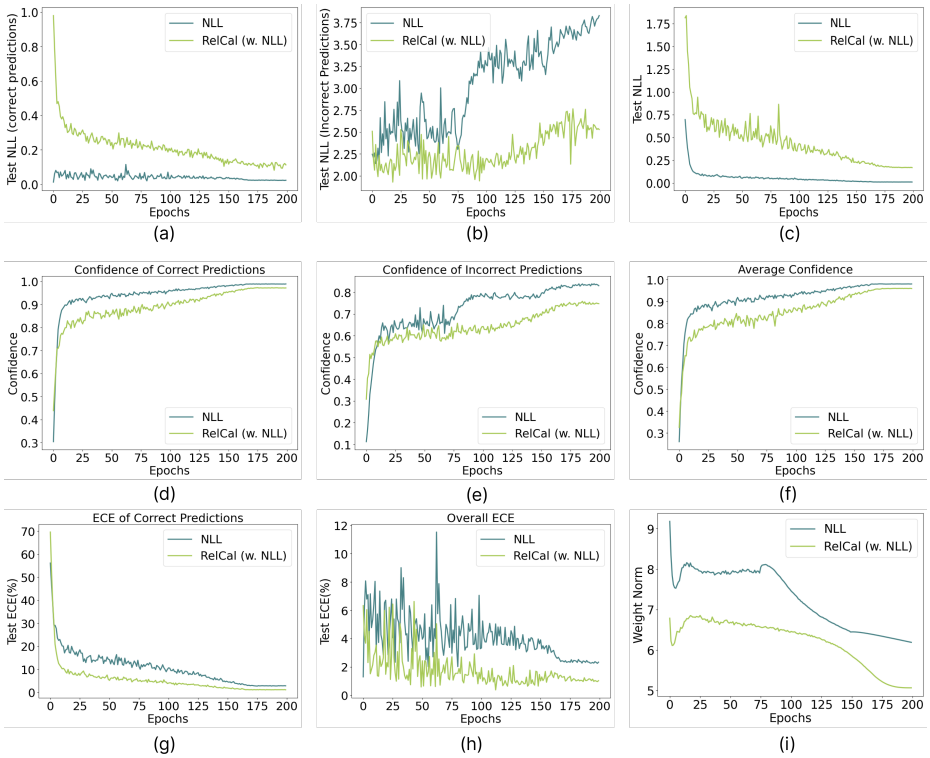


Fig. 2: We visualize ResNet-50's performance metrics on CIFAR-10 across training epochs.

2. `RelCal` **acts as weight regularizer.** Recall that the relevance matrix is defined as $\mathbf{R}[i,j] = r(h^{(i)}, j) = |h^{(i)}\mathbf{W}_{i,j}|$. Let $\mathbf{H} \in \mathbb{R}^{L\times L} = \text{diag}(h^{(1)}, h^{(2)}, \cdots, h^{(L)})$, be a diagonal matrix with the entries set as the feature embeddings. Then, the relevance matrix can be expressed as $\mathbf{R} = |\mathbf{HW}| = |\mathbf{H}||\mathbf{W}|$, where $|\cdot|$ represents the absolute value of the elements of the matrix and $\mathbf{W} \in \mathbb{R}^{L\times K}$ is the weight of the last linear layer. Based on this formulation, the logit output for the c-th class can also be written as:

$$\mathbf{f}_{\text{rel}}(\mathbf{x},\theta)[c] = \sum_{l=1}^{L}(\mathbf{R}\odot\mathbf{M})^{\top}[c,l] + \mathbf{b}[c], \tag{7}$$

$$= \sum_{l=1}^{L}(|\mathbf{H}||\mathbf{W}|\odot\mathbf{M})^{\top}[c,l] + \mathbf{b}[c], \tag{8}$$

$$= \sum_{l=1}^{L}((|\mathbf{W}|\odot\mathbf{M})^{\top}|\mathbf{H}|)[c,l] + \mathbf{b}[c], \tag{9}$$

$$= \sum_{l=1}^{L}(\mathbf{W}_{\text{rel}}^{\top}|\mathbf{H}|)[c,l] + \mathbf{b}[c], \text{ where } \mathbf{W}_{\text{rel}} = |\mathbf{W}|\odot\mathbf{M} \tag{10}$$

From Equation 2, model output based on standard training objective without `RelCal` can be written as:

$$\mathbf{f}(\mathbf{x}, \theta)[c] = \sum_{l=1}^{L} (\mathbf{W}^\top \mathbf{H})[c, l] + \mathbf{b}[c] \tag{11}$$

Comparing Equation 10 and Equation 11, we observe that training with `RelCal` provides an additional implicit regularization on the network weight $\mathbf{W}$. This observation is further supported by Fig. 2(i), where we plot the norm of the weight matrix $\|\mathbf{W}\|$ across different training epochs. We see that the weight norm of the `RelCal`-trained model is always lower than the model trained using only NLL loss, further indicating that `RelCal` effectively regularizes the model weights.

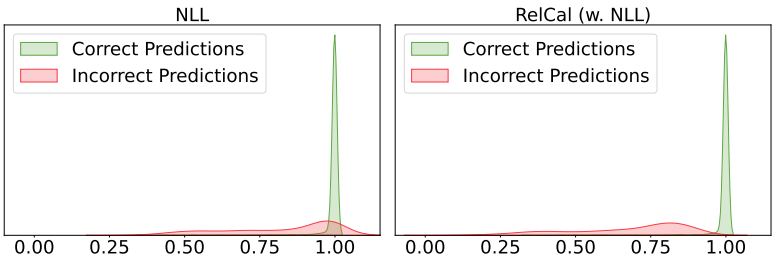
4.3 Exploring the Properties of `RelCal`

In this section, we provide a comprehensive understanding of the advantages offered by `RelCal` as well as explore its properties.

`RelCal` focuses on relevant cues. The core idea of our method is that learning class-relevant features can address the issue of overly confident predictions, leading to enhancement in model calibration. To examine our hypothesis, we analyse GradCAM [37] visualizations of few representative bird images from CUB-200-2011 [42] dataset in Fig. 4(b). To ensure a fair comparison, both models are trained using NLL loss. Our observations indicate that the model trained with `RelCal` exhibits a more precise focus on semantic attributes critical for accurate classification. For instance, the image of the "Winged Gull" in the last row has been misclassified by both models. However, it is noteworthy that the class predicted by the `RelCal`-trained model is much more relevant compared to the other model. Specifically, `RelCal` predicts the image as a "California Gull" which is closely related to the original image, whereas the other model outputs the class label as "Least Auklet" which is entirely unrelated to a Gull. This conclusion is further supported by the saliency maps, highlighting the enhanced discriminative capability of the `RelCal`-trained model.

`RelCal` reduces misclassification confidence. Fig. 3 shows distribution plots of confidence scores for correct and incorrect predictions made by a ResNet-50 model on the CIFAR-10 and CIFAR-10-LT dataset trained with and without `RelCal`. The results highlight that `RelCal` significantly reduces the average confidence level of incorrect predictions. This indicates that `RelCal` not only results in a more calibrated model but also provides a more realistic confidence measure for each prediction, particularly misclassification. To further corroborate the observation in Fig. 3, we validate our findings through an analysis in Fig. 4(a).

`RelCal` is model agnostic. To assess the model-agnostic nature of `RelCal`, we conducted experiments employing different ResNet architectures with varying capacities, ranging from 11.2M to 42.5M parameters. Specifically, we trained two models for each architecture—one using standard technique and other using `RelCal`. For training both models, we used focal loss [28]. The experimental results are presented in Table 6. We observe that training with `RelCal` consistently



(a) CIFAR-10

Fig. 3: Confidence score distribution for correct and misclassified predictions on CIFAR-10.

enhances model calibration, irrespective of the underlying model architecture. In Appendix, we provide further analysis utilizing other standard architectures such as WideResNet [44], and Vision Transformers [7].

Analysis on relevant feature selection. A key aspect of our algorithm is the strategic selection of the most relevant features for class predictions. In particular, the features are chosen based on the dimensions that contributed most to the class's output. In this analysis, we contrast our feature selection mechanism with a random feature selection method that depends on a stochastic process for randomly selecting elements within the feature vector. We report results of this analysis in Table 1. Empirical results highlight that randomly choosing feature dimensions is suboptimal for model calibration.

Method	**ECE** ↓	**AECE** ↓
Random features	8.74	8.97
Features with least relevance	11.73	12.90
`RelCal` (w. FL) (ours)	**2.45**	**2.43**

Table 1: **Analysis on relevant feature selection.** We compare different strategies for selecting the relevant features. All models are trained using focal loss [28]. Best performing results are marked in **bold**. Model is ResNet-50 and the dataset is CIFAR-100.

Training Time of `RelCal`. In Table 2, we compare the train time of `RelCal` against the standard training method using NLL loss for different datasets. We observe that training using `RelCal` incurs no additional computational overhead as compared to standard training procedures. Thus, our method not only alleviates miscalibration but also does not impose any extra training time.

Training Method	C-10	C-100	TIN-200
	(Train time in hours)		
NLL	3.27	3.11	7.30
`RelCal` (w. NLL) (Ours)	3.33	3.28	7.42

Table 2: **Computational cost for training**.We train ResNet-50 with the same batch size for both setups, using the software configuration reported in Appendix.

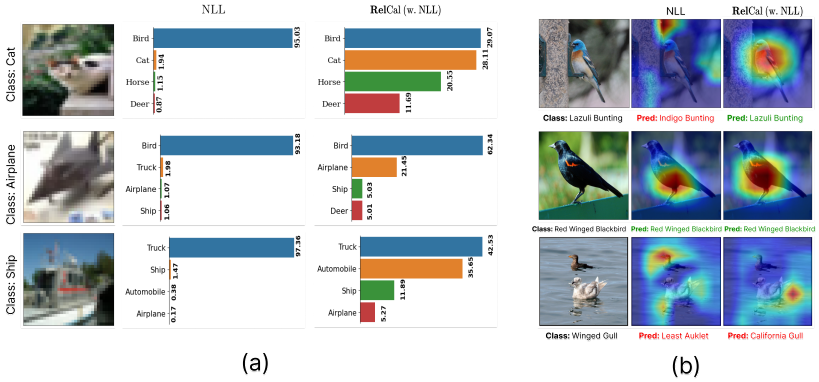


Fig. 4: (a) Training with `RelCal` reduces the confidence of the misclassifications. Images are from the CIFAR-10 dataset. (b) GradCAM visualization of models trained with and without `RelCal` using NLL loss. The model is ResNet-101 and the images are from CUB-200.

5 Experiments

In this section, we highlight the effectiveness of our proposed training approach `RelCal`. We present a set of experiments to evaluate `RelCal` against state-of-the-art train-time calibration methods on several benchmarks.

Datasets. To understand the effectiveness of `RelCal` for network calibration, we primarily evaluate `RelCal` on four standard image classification datasets - SVHN [31], CIFAR10 [18], CIFAR-100 [18], TinyImageNet-200 [23] (TIN-200) and one fine-grained image dataset CUB-200 [42]. We divide the training set into two distinct subsets: (a) a training subset comprising 90% of the samples, and (b) a validation subset with the remaining 10%, which is used for hyper-parameter optimization. Further, to analyse the generalization capability of our method, we also evaluate on the large-scale ImageNet [5] dataset. In addition, we also validate our technique on two imbalanced long-tailed datasets: CIFAR-10-LT [3], CIFAR-100-LT [3], and a standard natural language processing (NLP) dataset, Newsgroups [22]. Our choice of diverse datasets allows us to comprehensively assess the generalization capability of our proposed method, `RelCal`, for network calibration. For more detailed descriptions of each dataset, refer to the Appendix.

Evaluation Metrics. For evaluation, we report standard calibration metrics, including Expected Calibration Error (ECE), and Adaptive Expected Calibration Error (AECE), along with test accuracy to assess the calibration performance. For fair comparison, following previous studies [15, 27, 32], we use 15 bins to measure the ECE and AECE. Training with `RelCal` surpasses state-of-the-art calibration techniques, enhancing both calibration and test accuracy.

Training Details. For image-classification tasks, we used the ResNet [14] and WideResNet [44] architecture for primary experimentation. For the NLP classification task, following past studies [27], we train a Global Pooling CNN (GP-CNN) architecture [25]. The pruning percentile $p \in \{15, 25, 35, 55, 75, 85, 95\}$ is cross-validated as described in the Appendix. Following our validation strategy,

Dataset	Model	NLL			RelCal (w. NLL)			LS [29]			RelCal (w. LS [29])			FL [28]			RelCal (w. FL [28])		
		Acc	ECE	AECE	Acc	ECE	AECE	Acc	ECE	AECE	Acc	ECE	AECE	Acc	ECE	AECE	Acc	ECE	AECE
		↑	↓	↓	↑	↓	↓	↑	↓	↓	↑	↓	↓	↑	↓	↓	↑	↓	↓
CIFAR-10	R50	94.99	3.65	2.95	95.19	2.98	2.88	94.97	3.25	3.56	94.91	3.05	3.23	95.02	3.32	3.30	**95.32**	**0.97**	**1.02**
CIFAR-10	WRN-26-10	95.60	2.83	2.83	95.88	2.47	2.47	95.64	4.48	4.29	95.91	3.05	3.23	95.85	4.65	4.44	**96.08**	**0.60**	**0.80**
CIFAR-10-LT(IF=10)	R50	89.39	6.91	6.91	89.87	6.47	6.44	89.85	3.86	4.06	89.56	3.41	3.98	88.78	4.16	4.12	**89.95**	**3.21**	**3.21**
CIFAR-10-LT(IF=100)	R50	73.53	20.05	20.04	**76.19**	15.10	15.09	74.19	14.77	14.76	74.50	14.92	14.87	70.03	14.98	15.01	74.43	**14.71**	**14.67**
CIFAR-100	R50	78.13	12.54	12.39	**78.91**	9.34	9.23	77.27	6.87	6.84	78.54	3.92	3.83	77.69	5.51	5.40	78.32	**2.75**	**2.45**
CIFAR-100	WRN-26-10	80.21	8.29	8.34	**80.63**	6.50	6.44	79.27	2.94	2.93	80.32	3.12	3.59	80.42	4.62	4.73	80.10	**2.73**	**2.75**
CIFAR-100-LT(IF=10)	R50	60.53	13.23	13.14	**63.57**	12.57	12.43	62.31	6.73	6.75	61.88	6.18	6.12	62.09	6.16	6.19	62.40	**5.74**	**5.73**
CIFAR-100-LT(IF=100)	R50	39.00	32.13	32.12	40.60	28.39	28.38	39.26	19.98	19.94	**43.21**	20.91	20.93	36.20	20.76	20.75	41.52	**18.23**	**18.22**
TinyImagenet-200	R50	65.76	13.20	13.19	66.14	8.26	8.23	63.52	2.85	2.27	64.75	2.38	2.13	63.80	3.22	3.08	**66.48**	**2.17**	**2.19**
CUB-200	R101	72.57	18.19	18.18	72.85	8.46	8.38	73.25	12.51	12.52	**75.02**	3.45	3.46	72.87	8.41	8.39	73.11	**2.99**	**3.22**
SVHN	R50	96.15	2.41	2.40	**96.21**	2.10	2.05	95.83	4.35	4.18	96.20	4.60	4.38	96.14	4.16	4.21	96.10	**1.10**	**1.07**
20 Newsgroups	GP-CNN	66.89	22.74	22.78	65.07	22.06	22.47	66.55	6.67	6.36	66.78	**5.44**	**5.89**	67.03	13.32	13.30	**67.43**	10.88	10.86

Table 3: **Calibration performance with different loss functions.** Comparison of calibration performance when using RelCal with three commonly used loss functions (NLL/LS/FL). We observe that for all classification loss, training with RelCal enhances model calibration. Results obtained using RelCal are highlighted in blue. R50 represents ResNet-50 [14] model and WRN-26-10 represents WideResNet-26-10 [44] architecture. The best results for each dataset are marked in **bold**.

Methods	CIFAR-10 [18]						CIFAR-100 [18]						ImageNet [5]		
	ResNet-50			ResNet-101			ResNet-50			ResNet-101			ResNet-50		
	Acc	ECE	AECE	Acc	ECE	AECE	Acc	ECE	AECE	Acc	ECE	AECE	Acc	ECE	AECE
	↑	↓	↓	↑	↓	↓	↑	↓	↓	↑	↓	↓	↑	↓	↓
NLL	94.99	3.65	2.95	94.48	3.68	3.57	78.13	12.54	12.39	77.84	13.12	13.10	73.86	9.34	9.45
MMCE [21] (ICML 2018)	95.20	3.87	3.89	95.03	3.83	3.83	77.12	12.90	12.89	77.56	13.53	13.39	74.29	8.81	8.83
LS [29] (NeurIPS 2019)	94.97	3.25	3.56	94.10	3.21	3.21	77.27	6.87	6.84	76.98	8.54	8.58	75.14	3.31	3.23
FL [28] (NeurIPS 2020)	95.02	3.32	3.30	95.16	3.48	3.49	77.69	5.51	5.40	77.12	4.99	5.01	74.69	3.93	3.94
FL+MDCA [15] (CVPR 22)	95.16	1.84	1.78	95.18	2.01	2.15	75.46	5.71	5.71	77.21	3.74	3.79	75.05	6.95	6.33
CPC [4] (CVPR 22)	95.10	4.67	4.67	**95.38**	5.34	5.39	77.78	10.98	10.90	77.82	12.17	12.18	74.98	4.38	4.32
MbLS [27] (CVPR 22)	95.24	1.21	2.98	95.31	**1.39**	3.45	77.12	4.56	4.56	77.55	5.48	5.78	75.29	4.24	4.28
RankMixup [32] (ICCV 23)	94.88	2.59	2.58	94.25	3.24	3.21	77.11	3.46	3.45	76.46	3.49	3.49	74.86	3.93	3.92
RelCal (w. FL) (Ours)	**95.32**$_{\pm 0.49}$	**0.97**$_{\pm 0.18}$	**1.02**$_{\pm 0.16}$	**95.38**$_{\pm 0.36}$	1.41$_{\pm 0.20}$	**1.40**$_{\pm 0.17}$	**78.32**$_{\pm 0.36}$	**2.75**$_{\pm 0.40}$	**2.45**$_{\pm 0.37}$	**78.12**$_{\pm 0.44}$	**2.43**$_{\pm 0.28}$	**2.44**$_{\pm 0.36}$	**75.51**$_{\pm 0.75}$	**3.06**$_{\pm 0.43}$	**3.08**$_{\pm 0.38}$

Table 4: **Comparison with state-of-art.** We compare RelCal with an array of state-of-art train-time calibration techniques on CIFAR10/100 [18] and ImageNet [5] dataset. Best results are highlighted in **bold**. All values are in percentage (%). Mean and std of our method are estimated over 3 random runs.

we set $p = 85$ for the ResNet-50 model and $p = 95$ for the ResNet-101 and WideResNet-26-10 model across all datasets. We report ablation results for the effect of p in Section 7.2. We provide further training details in Appendix.

Baselines. We compare our method against an array of competitive train-time calibration techniques including Negative Log Likelihood (NLL), Label Smoothing [29], MMCE [21], Focal Loss (FL) [28], FL+MDCA [15], CPC [4], MbLS [27] and RankMixup [32]. For detailed description of each method, refer Appendix.

6 Results

Investigating the impact of different loss functions with RelCal. Since RelCal is designed to learn features relevant to each class, it can be seamlessly integrated with standard classification and calibration losses such as Negative Log Likelihood (NLL), Label Smoothing (LS) [29], Focal Loss (FL) [28], MDCA [15], or MbLS [27]. Due to space constraints, we show comparative analysis of model calibration using NLL, LS, and FL in Table 3. We report additional results using MDCA and MbLS in Appendix. Our empirical evaluation reveals several key insights: **(1.)** Integrating RelCal with these loss functions substantially enhances

Methods	ResNet-50 [14]					
	CIFAR-10-LT (IF=10)			CIFAR-100-LT (IF=10)		
	Acc	ECE	AECE	Acc	ECE	AECE
	↑	↓	↓	↑	↓	↓
NLL	89.39	6.91	6.91	60.53	13.23	13.14
LS [29]	89.85	3.86	4.06	62.31	6.73	6.75
FL [28]	88.78	4.16	4.12	62.09	6.16	6.19
FL+MDCA [15]	87.60	5.45	5.44	46.20	11.32	11.43
MbLS [27]	87.82	7.04	6.99	58.10	8.36	8.93
RankMixup [32]	89.80	5.80	5.84	**63.83**	10.01	9.98
`RelCal` (w. FL)(Ours)	**89.95**	**3.21**	**3.21**	62.40	**5.74**	**5.73**

Table 5: **Results on Imbalanced Datasets.** Comparison of calibration performances on two imbalanced datasets CIFAR-10-LT [3] and CIFAR-100-LT [3]. For imbalanced datasets, we use an imbalance factor (IF) of 10.

Model	Params	FLOPS	Optimal p	ECE (%) (↓)		
				FL	`RelCal` (w.FL)	Δ
R-18	11.2M	0.6G	15	1.11	0.63	+0.48
R-34	21.2M	1.1G	65	2.69	1.59	+1.10
R-50	23.5M	1.3G	85	3.32	0.97	+2.35
R-101	42.5M	2.5G	95	3.48	1.41	+2.07

Table 6: **Exploring relationship between model capacity and pruning percentile** p. We observe training with `RelCal` leads to more gains in calibration for models with larger capacities.

calibration performance across various datasets and architectures. For instance, employing `RelCal` in conjunction with NLL loss on the TIN-200 dataset leads to a reduction in both ECE and AECE by 4.94% and 4.96% respectively, as compared to the standard NLL training. **(2.)** Beyond calibration improvements, `RelCal` also enhances test accuracy. Notably, when used in conjunction with FL, we see a 5.32% improvement in accuracy on the CIFAR-100-LT dataset. **(3.)**Out of all configurations tested, the combination of `RelCal` and FL [28] demonstrates superior calibration performance across most datasets.

Based on these observations, for the subsequent sections of this paper, we primarily train `RelCal` in conjunction with Focal Loss [28] unless stated otherwise.

Comparison with state-of-the-art. Table 4 compares our `RelCal` with an array of competitive train-time calibration techniques. We observe that `RelCal` consistently improves ECE and AECE on all the datasets and different architectures. Notably, on CIFAR-10 with ResNet-50 architecture, `RelCal` reduces the ECE to **0.97**% and AECE to **1.02**%, which are considerable improvements over state-of-art techniques such as RankMixup [32] and MbLS [27]. Further, we observe that `RelCal` also demonstrates superior performance on the large-scale ImageNet [5] dataset thereby highlighting the importance of learning class-relevant features. Further, we also note that `RelCal` is able to achieve the best calibration performance without sacrificing on the test accuracy.

Performance on imbalanced datasets. In Table 4, we highlighted the impressive performance of `RelCal` over various standard datasets. However, it is essential to recognize that these datasets are predominantly balanced, which may not fully capture the challenges that models face in real-world scenarios. When deployed in the wild, a model is more likely to encounter skewed and long-tailed distributions where few classes dominate over the rare classes [36]. To simulate this setup, following [3], we deliberately introduce class imbalance to the CIFAR dataset to create a long-tail (-LT) distribution. This type of evaluation is important for understanding how well the model can make confident

and reliable predictions, especially for rare classes, where class distributions are often imbalanced. The results of this experiment are presented in Table 5, where it is noteworthy that `RelCal` achieves the highest calibration performance for both CIFAR-10-LT and CIFAR-100-LT datasets.

Performance on different architectures. In Table 4, we established the superiority of `RelCal` on ResNet [14]. Going beyond, in Table 7, we show that `RelCal` remains competitive and outperforms the state-of-art for other common architectures such as WideResNet [44]. From Table 7, we observe that on CIFAR-10 dataset using WideResNet-26-10 model, `RelCal` reduces the ECE to 0.60%– a 1.12% improvement over RankMixup [32].

Methods	WideResNet-26-10 [44]					
	CIFAR-10 [18]			CIFAR-100 [18]		
	Acc	ECE	AECE	Acc	ECE	AECE
	↑	↓	↓	↑	↓	↓
NLL	95.60	2.83	2.83	80.21	8.29	8.34
LS [29]	95.64	4.48	4.29	79.27	2.94	2.93
FL [28]	95.85	4.65	4.44	**80.42**	4.62	4.73
FL+MDCA [15]	95.90	0.98	1.27	80.05	3.13	3.19
MbLS [27]	95.89	1.32	2.56	79.21	4.58	4.51
RankMixup [32]	95.88	1.72	1.49	78.56	3.13	3.24
`RelCal` (w. FL)(Ours)	**96.08**	**0.60**	**0.80**	80.10	**2.73**	**2.75**

Table 7: Results using WideResNet architecture.

7 Discussion

7.1 Exploring relationship model capacity and p

In this section, we investigate the presence of any correlation between model capacity and the optimal pruning percentile p. To this end, we utilize a series of ResNet [14] models with varying capacities, specifically ResNet-18, ResNet-34, ResNet-50, and ResNet-101. We present the results of our experiments in Table 6. Our findings reveal two important insights: (1) We discern a positive correlation between the pruning percentile p and number of model parameters. This observation aligns with the analysis presented in [11], which demonstrated that models with larger capacities are more susceptible to overfitting. (2) The advantages gained from training with `RelCal` become notably more pronounced in models with larger capacities, such as ResNet-50 and ResNet-101.

7.2 Ablations on pruning percentile p

In this ablation, we aim to understand the effect of pruning percentile p on the calibration performance. The pruning percentile p plays a critical role in determining the threshold for selecting the relevant features during training. A higher value of p implies a more stringent criterion for feature relevance, potentially pruning more connections deemed less significant for each class. This ablation is based on a ResNet-50 model trained on CIFAR-10 dataset. Specifically, we train and compare multiple models by varying $p \in \{15, 25, 35, 55, 75, 85, 95\}$. Fig. 5 reports the influence of p on the Expected Calibration Error (ECE). We observe that: (1) Setting $p = 85$ provides the optimal calibration performance, which is consistent

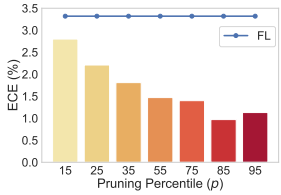


Fig. 5: Effect of varying the pruning percentile (p) on calibration performance of `RelCal` (w. FL).

with one chosen using our validation strategy (see Appendix). Further, irrespective of the pruning percentile used, `RelCal` is consistently better than FL [28]. (2) Lower p values significantly diminish the model's calibration performance, corroborating the necessity of pruning irrelevant features. (3) In the extreme case, when p is too high (*e.g.* $p = 95$), we observe a slight deterioration in the calibration performance.

Additional Comparison. In Appendix, we also compare `RelCal` with other common sparsification techniques like Unit Dropout [39], Targeted Dropout [9], DICE [41], Adaptive Dropout [2], and Guided Dropout [16]. Our findings show that `RelCal` consistently achieves the best ECE.

8 Conclusion

We propose `RelCal`, a simple and novel training approach designed to enhance network calibration by concentrating on class-relevant features. Our comprehensive experiments across various datasets and architectures demonstrate the effectiveness of `RelCal` in improving model calibration without much increase in training time. Notably, training `RelCal` in conjunction with focal loss consistently outperforms state-of-art train-time calibration methods, achieving significant reductions in both ECE and AECE metrics. Furthermore, our approach also exhibits promising results on challenging long-tailed datasets. Our mathematical analysis suggests that `RelCal` enhances model calibration by implicitly regularizing network weights, thereby mitigating overconfident predictions.

References

1. Ahn, Y.H., Park, G.M., Kim, S.T.: Line: Out-of-distribution detection by leveraging important neurons (2023) 4
2. Ba, J., Frey, B.: Adaptive dropout for training deep neural networks. Advances in neural information processing systems **26** (2013) 4, 14
3. Cao, K., Wei, C., Gaidon, A., Arechiga, N., Ma, T.: Learning imbalanced datasets with label-distribution-aware margin loss. In: Advances in Neural Information Processing Systems (2019) 10, 12
4. Cheng, J., Vasconcelos, N.: Calibrating deep neural networks by pairwise constraints. In: 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 13699–13708 (2022). https://doi.org/10.1109/CVPR52688.2022.01334 3, 11
5. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition. pp. 248–255. Ieee (2009) 10, 11, 12
6. Djurisic, A., Bozanic, N., Ashok, A., Liu, R.: Extremely simple activation shaping for out-of-distribution detection. In: The Eleventh International Conference on Learning Representations (2023), https://openreview.net/forum?id=ndYXTEL6cZz 4
7. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N.: An image is worth 16x16 words: Transformers for image recognition at scale. In: International Conference on Learning Representations (2021) 9

8. Dusenberry, M.W., Tran, D., Choi, E., Kemp, J., Nixon, J., Jerfel, G., Heller, K., Dai, A.M.: Analyzing the role of model uncertainty for electronic health records. Proceedings of the ACM Conference on Health, Inference, and Learning (2020) 1
9. Gomez, A.N., Zhang, I., Kamalakara, S.R., Madaan, D., Swersky, K., Gal, Y., Hinton, G.E.: Learning sparse networks using targeted dropout. arXiv preprint arXiv:1905.13678 (2019) 14
10. Grigorescu, S., Trasnea, B., Cocias, T., Macesanu, G.: A survey of deep learning techniques for autonomous driving. Journal of Field Robotics **37**(3), 362–386 (2020) 1
11. Guo, C., Pleiss, G., Sun, Y., Weinberger, K.Q.: On calibration of modern neural networks. In: International Conference on Machine Learning. pp. 1321–1330. PMLR (2017) 1, 2, 3, 4, 6, 13
12. Han, S., Mao, H., Dally, W.J.: Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding (2016) 4
13. Han, S., Pool, J., Tran, J., Dally, W.: Learning both weights and connections for efficient neural network. In: Advances in neural information processing systems. pp. 1135–1143 (2015) 4
14. He, K., Zhang, X., Ren, S., Sun, J.: Identity mappings in deep residual networks (2016) 3, 10, 11, 12, 13
15. Hebbalaguppe, R., Prakash, J., Madan, N., Arora, C.: A stitch in time saves nine: A train-time regularizing loss for improved neural network calibration. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 16081–16090 (2022) 2, 3, 10, 11, 12, 13
16. Keshari, R., Singh, R., Vatsa, M.: Guided dropout. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 33, pp. 4065–4072 (2019) 14
17. Kriegeskorte, N.: Deep neural networks: a new framework for modeling biological vision and brain information processing. Annual review of vision science **1**, 417–446 (2015) 2
18. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images. Tech. Rep. 0, University of Toronto, Toronto, Ontario (2009) 10, 11, 13
19. Kull, M., Perello-Nieto, M., Kängsepp, M., Song, H., Flach, P., et al.: Beyond temperature scaling: Obtaining well-calibrated multiclass probabilities with dirichlet calibration. arXiv preprint arXiv:1910.12656 (2019) 4
20. Kull, M., Silva Filho, T., Flach, P.: Beta calibration: a well-founded and easily implemented improvement on logistic calibration for binary classifiers. In: Artificial Intelligence and Statistics. pp. 623–631. PMLR (2017) 4
21. Kumar, A., Sarawagi, S., Jain, U.: Trainable calibration measures for neural networks from kernel mean embeddings. In: International Conference on Machine Learning. pp. 2805–2814. PMLR (2018) 3, 11
22. Lang, K.: Newsweeder: Learning to filter netnews. In: Machine Learning Proceedings 1995, pp. 331–339. Elsevier (1995) 10
23. Le, Y., Yang, X.: Tiny imagenet visual recognition challenge. CS 231N **7**(7), 3 (2015) 10
24. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. nature **521**(7553), 436–444 (2015) 2
25. Lin, M., Chen, Q., Yan, S.: Network in network. arXiv preprint arXiv:1312.4400 (2013) 10
26. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. In: Proceedings of the IEEE international conference on computer vision. pp. 2980–2988 (2017) 2, 3

27. Liu, B., Ben Ayed, I., Galdran, A., Dolz, J.: The devil is in the margin: Margin-based label smoothing for network calibration. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 80–88 (2022) 3, 10, 11, 12, 13
28. Mukhoti, J., Kulharia, V., Sanyal, A., Golodetz, S., Torr, P., Dokania, P.: Calibrating deep neural networks using focal loss. Advances in Neural Information Processing Systems **33**, 15288–15299 (2020) 1, 2, 3, 6, 8, 9, 11, 12, 13, 14
29. Müller, R., Kornblith, S., Hinton, G.E.: When does label smoothing help? Advances in neural information processing systems **32** (2019) 1, 2, 3, 11, 12, 13
30. Naeini, M.P., Cooper, G., Hauskrecht, M.: Obtaining well calibrated probabilities using bayesian binning. In: Twenty-Ninth AAAI Conference on Artificial Intelligence (2015) 2, 4
31. Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., Ng, A.Y.: Reading digits in natural images with unsupervised feature learning (2011) 10
32. Noh, J., Park, H., Lee, J., Ham, B.: Rankmixup: Ranking-based mixup training for network calibration. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 1358–1368 (2023) 1, 10, 11, 12, 13
33. Park, H., Noh, J., Oh, Y., Baek, D., Ham, B.: Acls: Adaptive and conditional label smoothing for network calibration. In: Proceedings of the IEEE/CVF ICCV (2023) 3
34. Pereyra, G., Tucker, G., Chorowski, J., Kaiser, Ł., Hinton, G.: Regularizing neural networks by penalizing confident output distributions. arXiv preprint arXiv:1701.06548 (2017) 3
35. Platt, J., 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, 4
36. Reed, W.J.: The pareto, zipf and other power laws. Economics letters **74**(1), 15–19 (2001) 12
37. Selvaraju, R.R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., Batra, D.: Grad-cam: Visual explanations from deep networks via gradient-based localization. In: 2017 IEEE International Conference on Computer Vision (ICCV). pp. 618–626 (2017) 8
38. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014) 5
39. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research **15**(1), 1929–1958 (2014) 4, 14
40. Sun, Y., Guo, C., Li, Y.: React: Out-of-distribution detection with rectified activations. Advances in NeurIPS **34**, 144–157 (2021) 4
41. Sun, Y., Li, Y.: Dice: Leveraging sparsification for out-of-distribution detection. In: Proceedings of European Conference on Computer Vision (2022) 4, 14
42. Wah, C., Branson, S., Welinder, P., Perona, P., Belongie, S.: The caltech-ucsd birds-200-2011 dataset (2011) 3, 8, 10
43. Zadrozny, B., Elkan, C.: Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers. In: Proceedings of the International Conference on Machine Learning. vol. 1, pp. 609–616 (2001) 2, 4
44. Zagoruyko, S., Komodakis, N.: Wide residual networks. BMVC (2016) 3, 9, 10, 11, 13
45. Zhang, H., Cisse, M., Dauphin, Y.N., Lopez-Paz, D.: mixup: Beyond empirical risk minimization. In: ICLR (2018) 3