Class Adaptive Network Calibration

Bingyuan Liu[*1], Jérôme Rony[*1], Adrian Galdran[2], Jose Dolz[1], Ismail Ben Ayed[1]

[1]ÉTS Montreal, Canada   [2]Universitat Pompeu Fabra, Barcelona, Spain

Abstract

Recent studies have revealed that, beyond conventional accuracy, calibration should also be considered for training modern deep neural networks. To address miscalibration during learning, some methods have explored different penalty functions as part of the learning objective, alongside a standard classification loss, with a hyper-parameter controlling the relative contribution of each term. Nevertheless, these methods share two major drawbacks: 1) the scalar balancing weight is the same for all classes, hindering the ability to address different intrinsic difficulties or imbalance among classes; and 2) the balancing weight is usually fixed without an adaptive strategy, which may prevent from reaching the best compromise between accuracy and calibration, and requires hyper-parameter search for each application. We propose Class Adaptive Label Smoothing (CALS) for calibrating deep networks, which allows to learn class-wise multipliers during training, yielding a powerful alternative to common label smoothing penalties. Our method builds on a general Augmented Lagrangian approach, a well-established technique in constrained optimization, but we introduce several modifications to tailor it for large-scale, class-adaptive training. Comprehensive evaluation and multiple comparisons on a variety of benchmarks, including standard and long-tailed image classification, semantic segmentation, and text classification, demonstrate the superiority of the proposed method. The code is available at https://github.com/by-liu/CALS *.*

1. Introduction

Deep Neural Networks (DNNs) have become the prevailing model in machine learning, particularly for computer vision [13] and natural language processing applications [44]. Increasingly powerful architectures [3,13,24], learning methods [4,12] and a large body of other techniques [15,27] are constantly introduced. Nonetheless, recent studies [11,31] have shown that regardless of their superior discriminative

performance, high-capacity modern DNNs are poorly calibrated, *i.e.* failing to produce reliable predictive confidences. Specifically, they tend to yield over-confident predictions, where the probability associated with the predicted class overestimates the actual likelihood. Since this is a critical issue in safety-sensitive applications like autonomous driving or computational medical diagnosis, the problem of DNN calibration has been attracting increasing attention in recent years [11,31,38].

Current calibration methods can be categorized into two main families. The first family involves techniques that perform an additional post-processing parameterized operation on the output logits (or pre-softmax activations) [11], with the calibration parameters of that operation obtained from a validation set by either learning or grid-search. Despite the simplicity and low computational cost, these methods have empirically proven to be highly effective [8,11]. However, their main drawback is that the choice of the optimal calibration parameters is highly sensitive to the trained model instance and validation set [22,31].

The second family of methods attempts to simultaneously optimize for accuracy and calibration during network training. This is achieved by introducing, explicitly or implicitly, a secondary optimization goal involving the model's predictive uncertainty, alongside the main training objective. As a result, a scalar balancing hyper-parameter is required to tune the relative contribution of each term in the overall loss function. Some examples of this type of approaches include: Explicit Confidence Penalty (ECP) [38], Label Smoothing (LS) [32], Focal Loss (FL) [21] and its variant, Sample-Dependent Focal Loss (FLSD) [31]. It has been recently demonstrated in [22] that all these methods can be formulated as different penalty terms that enforce the same equality constraint on the logits of the DNN: driving the *logit distances* towards zero. Here, logit distances refers to the vector of L1 distances between the highest logit value and the rest. Observing the non-informative nature of this equality constraint, [22] proposed to use a generalized inequality constraint, only penalizing those logits for which the distance is larger than a pre-defined margin, achieving state-of-the-art calibration performance on many different benchmarks.

Although learning based methods achieve greater calibra-

---

[*]Equal Contributions. Correspondence to: {liubingyuan1988@gmail.com, jerome.rony.1@etsmtl.net}
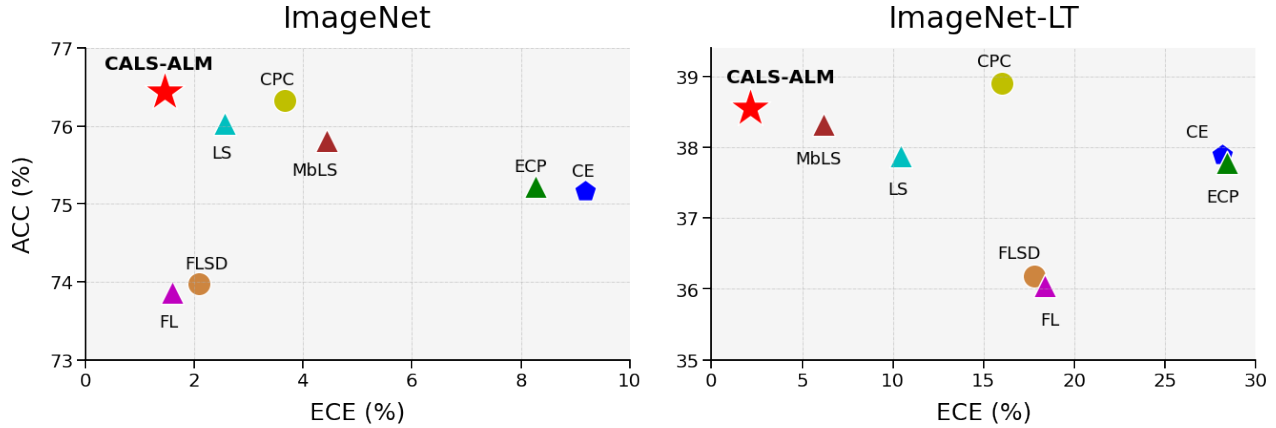
Figure 1. Many techniques have been proposed for jointly improving accuracy and calibration during training [11,31], but they fail to consider uneven learning scenarios like high class imbalance or long-tail distributions. We show a comparison of the proposed **CALS-ALM** method and different learning approaches in terms of Calibration Error (ECE) vs Accuracy on the (a) ImageNet and (b) ImageNet-LT (long-tailed ImageNet) datasets. A lower ECE indicates better calibration: a better model should attain **high ACC and low ECE**. Among all the considered methods, **CALS-ALM** shows superior performance when considering both discriminative power and well-balanced probabilistic predictions, achieving best accuracy and calibration on ImageNet, and best calibration and second best accuracy on ImageNet-LT.

tion performance [22,31], they have two major limitations: 1) The scalar balancing weight is equal for all classes. This hinders the network performance when some classes are harder to learn or less represented than others, such as in datasets with a large number of categories (ImageNet) or considerable class imbalance (ImageNet-LT). 2) The balancing weight is usually fixed before network optimization, with no learning or adaptive strategy throughout training. This can prevent the model from reaching the best compromise between accuracy and calibration. To address the above issues, we introduce Class Adaptive Label Smoothing method based on an Augmented Lagrangian Multiplier algorithm, which we refer to as CALS-ALM. Our **Contributions** can be summarized as follows:

- We propose Class Adaptive Label Smoothing (CALS) for network calibration. Adaptive class-wise multipliers are introduced instead of the widely used single balancing weight, which addresses the above two issues: 1) CALS can handle a high number of classes with different intrinsic difficulties, *e.g.* ImageNet; 2) CALS can effectively learn from data suffering from class imbalance or a long-tailed distribution, *e.g.* ImageNet-LT.

- Different from previous penalty based methods, we solve the resulting constrained optimization problem by implementing a modified Augmented Lagrangian Multiplier (ALM) algorithm, which yields adaptive and optimal weights for the constraints. We make some critical design decisions in order to adapt ALM to the nature of modern learning techniques: 1) The inner convergence criterion in ALM is relaxed to a fixed number of iterations in each inner stage, which is amenable to mini-batch stochas-

tic gradient optimization in deep learning. 2) Popular techniques, such as data augmentation, batch normalization [15] and dropout [10], rule out the possibility of tracking original samples and applying sample-wise multipliers. To overcome this complication, we introduce class-wise multipliers, instead of sample-wise multipliers in the standard ALM. 3) The outer-step update for estimating optimal ALM multipliers is performed on the validation set, which is meaningful for training on large-scale training set and avoids potential overfitting.

- Comprehensive experiments over a variety of applications and benchmarks, including standard image classification (Tiny-ImageNet and ImageNet), long-tailed image classification (ImageNet-LT), semantic segmentation (PASCAL VOC 2012), and text classification (20 Newsgroups), demonstrate the effectiveness of our CALS-ALM method. As shown in Figure 1, CALS-ALM yields superior performance over baselines and state-of-the-art calibration losses when considering both accuracy and calibration, especially for more realistic large-scaled datasets with large number of classes or class imbalance.

2. Related Work

2.1. Problem Formulation

Given a dataset $\mathcal{D} = \{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$ with $N \in \mathbb{N}$ pairs of samples $\boldsymbol{x} \in \mathcal{X}$ and corresponding labels $y \in \mathcal{Y}$, with $\mathcal{Y} = \{1, \ldots, K\}$, a deep neural network (DNN) $F_\theta : \mathcal{X} \rightarrow \mathbb{R}^K$ parameterized by θ yields *logits* $\boldsymbol{l} = F_\theta(\boldsymbol{x}) = (F_\theta(\boldsymbol{x})_k)_{1 \leq k \leq K} \in \mathbb{R}^K$. In a classification scenario, the output probability $\boldsymbol{s} = (s_k)_{1 \leq k \leq K} \in \Delta^{K-1}$, where $\Delta^{K-1} \subset [0, 1]^K$ denotes the probability simplex, is

obtained by applying the $\mathrm{softmax}$ function on the output *logits*, *i.e.* $\boldsymbol{s} = \mathrm{softmax}(\boldsymbol{l}) = \frac{\exp \boldsymbol{l}}{\sum \exp \boldsymbol{l}}$. Therefore, the predicted class $\hat{y}$ is computed as $\hat{y} = \arg\max_k s_k$, and the predicted confidence is $\hat{p} = \boldsymbol{s}_{\hat{y}} = \max_k s_k$. A perfectly calibrated model should satisfy that the predicted confidence of any input is equal to the accuracy of the model: $\hat{p} = \mathbb{P}(\hat{y} = y | \hat{p})$. Hence, an over-confident model yields on average larger confidences than the associated accuracy, whereas an under-confident model yields lower confidence than its accuracy.

A number of recent studies [11, 22, 30, 31] have shown that DNNs tend to become over-confident during training as a result of minimizing the popular cross-entropy (CE) training loss:

$$\mathcal{L}_{\mathrm{CE}}(\boldsymbol{x}, y) = -\sum_{k=1}^{K} y_k \log s_k \tag{1}$$

where $\boldsymbol{y} \in \{0, 1\}^K$ is the one-hot encoding of y. This objective function is minimized when the predictions for all the training samples fully match the ground-truth labels y, *i.e.* $s_y = 1$ and $\forall k \neq y, s_k = 0$. The negative logarithmic term on the logit of the correct category renders the global minimization of the CE loss unreachable, as it keeps pushing the predicted probabilities $\boldsymbol{s}$ towards the vertices of the $(K-1)$-simplex even after the classification error is zero [31], resulting in over-confident models.

2.2. Post-processing methods

To address mis-calibration, different post-processing techniques applied after model training have been proposed recently [8, 11, 43]. The most popular of these strategies is Temperature Scaling (TS) [11], which applies a single scalar temperature parameter to manipulate logit outputs monotonely, resulting in softened prediction confidences without affecting predicted labels. Note that here the temperature parameter needs to be tuned on a separate validation set. Despite its simplicity, TS has been shown effective in fixing over-confidence predictions [11]. As a local alternative to TS, [8] proposed to train a regression model for learning position-specific temperature for semantic segmentation problems. Unfortunately, TS and its variants can be sensitive to both the model and the validation set, and do not work well under data distribution shifts [36]. Thus, some subsequent works [28, 43] have attempted to provide solutions for improving performance under domain shift.

2.3. Learning-based methods

Another popular direction is to directly address mis-calibration during training by introducing an additional penalty or supervision regarding model calibration with the standard training loss. In [18], the authors introduced a trainable calibration measure based on RKHS kernels, while [16]

proposed a differential calibration loss based on a soft version of the binning operation in the ECE metric. In [5], two types of binary pairwise calibration constraints were proposed as additional penalty terms during training. Other methods try to decrease over-fitting on the cross-entropy loss, which has been demonstrated to be the main reason of over-confidence [11, 31]. In [38] an explicit confidence penalty (ECP) is proposed to maximize the entropy and reduce over-fitting, while Label Smoothing [42] has also been shown to implicitly improve the calibration [32] by softening the hard one-hot targets in the cross-entropy. The Focal Loss [21], originally proposed to tackle class imbalance, can also be effective for calibration [31], as it implicitly minimizes the Kullback-Leibler (KL) divergence between the uniform distribution and the network softmax probabilities, thereby increasing the entropy of the predictions. As an extension the Sample-Dependent Focal Loss (FLSD) was also proposed in [31] to further boost calibration performance.

Margin-based Label Smoothing (MbLS). A unifying constrained-optimization formulation of loss functions promoting calibration has been recently presented in [22]. Specifically, the additional penalties integrated in these methods, including ECP [38], LS [32] and FL [31], can be viewed as different forms of approximations to the same constraint, *i.e.* enforcing the logit distances to be zero. Noticing that this constraint is non-informative (its solution being uniformly distributed probabilities), [22] further proposed a generalized formulation by relaxing the constraint to allow the logit distances being lower than a strictly positive margin.

The specific formulation of MbLS [22] is as follows. Given a margin $m \in \mathbb{R}_+$, the constrained optimization problem for network training is:

$$\begin{aligned} \min_{\theta} \quad & \sum_{i=1}^{N} \mathcal{L}_{\mathrm{CE}}(\boldsymbol{x}^{(i)}, y^{(i)}) \\ \mathrm{s.\,t.} \quad & \max_{k}\{\boldsymbol{l}_k^{(i)}\} - l^{(i)} \preceq m\mathbf{1}_K, \quad i = 1, \ldots, N, \end{aligned} \tag{2}$$

where $\boldsymbol{l}^{(i)} = F_\theta(\boldsymbol{x}^{(i)})$. The minimum can be approximated by penalty-based optimization methods, transforming the above formulation into an unconstrained problem by means of simple ReLU functions:

$$\begin{aligned} \min_{\theta} \quad & \sum_{i=1}^{N} \mathcal{L}_{\mathrm{CE}}(\boldsymbol{x}^{(i)}, y^{(i)}) + \\ & \lambda \sum_{i=1}^{N} \sum_{j=1}^{K} \max\{0, \max_k\{l_k^{(i)}\} - l_j^{(i)} - m\}, \end{aligned} \tag{3}$$

where $\lambda \in \mathbb{R}_+$ is a scalar weight balancing contributions of the CE loss and the corresponding penalty.

3. Sample-wise Constrained DNN Optimization

Although MbLS can significantly improve calibration, the associated constrained problem (2) is not solved accurately. It is approximated by an unconstrained problem with a single uniform penalty, regardless of the data sample or category. However, the samples and classes considered in a classification problem have different intrinsic learning difficulties. Therefore, an improved training scheme would involve considering distinct penalty weights λ for each sample and class. This would result in having to chose $N \times K$ penalty weights $\Lambda \in \mathbb{R}_+^{N \times K}$, with the resulting optimization problem being:

$$
\min_{\theta} \quad \sum_{i=1}^{N} \mathcal{L}_{\text{CE}}(\boldsymbol{x}^{(i)}, y^{(i)}) + \\
\sum_{i=1}^{N} \sum_{j=1}^{K} \Lambda_{ij} \max\{0, \max_{k}\{l_k^{(i)}\} - l_j^{(i)} - m\}. \quad (4)
$$

From an optimization perspective, supposing that optimal weights $\theta^{\star}$ exists for problem (2), there exists $\Lambda^{\star} \in \mathbb{R}_+^{N \times K}$ such that $(\theta^{\star}, \Lambda^{\star})$ is a saddle point of the Lagrangian associated to (2). These $\Lambda^{\star}$ are the Lagrange multipliers of the problem. Therefore, using $\Lambda = \Lambda^{\star}$ would be the best choice to solve (4).

In practice, using the Lagrange multipliers of problem (2) as the weights for the penalties may not be computationally feasible, and it could even result in degraded performance. Indeed, in the context of machine learning, we optimize a model's weights θ to solve (2) on a training set $\mathcal{D}_{\text{train}}$, and expect to generalize on a test set $\mathcal{D}_{\text{test}}$, which we do not have access to during training. Because of the bias-variance trade-off, solving (2) optimally would likely result in overfitting, *i.e.* we may solve problem (2) accurately on the train set, but not generalize properly on the test set, resulting in poor calibration and classification performance overall. This suggests that it could be preferable to evaluate during training the quality of multipliers on a separate validation set $\mathcal{D}_{\text{val}}$. Additionally, several mechanisms for training DNNs are not compatible with a straightforward minimization. First, the use of batch normalization yields predictions that are not independent between samples in a minibatch. Second, the use of regularization techniques such as dropout may lead to virtually inaccurate predictions on certain training samples, impacting the correct estimation of multipliers. Third, data augmentation, which is standard in DNN training, would result in additional penalty weights for the augmented samples: they can be easier or harder to classify than the original ones.

In addition to the above obstacles, applying a penalty weight per sample and per class (resulting in $N \times K$ weights) would not scale well for large datasets and dense predictive tasks, such as semantic segmentation, which is typically formulated as a per-pixel classification task. Assuming that im-

ages in the dataset have a size of $H \times W$, this would result in $N \times H \times W \times K$ penalty weights. This rapidly becomes a limiting factor for moderately sized segmentation datasets. For instance, Pascal VOC 2012 [9] contains 21 classes and 1464 training images, amounting to 2.62×10^8 pixels, or 5.5×10^9 penalty weights, which, stored as `float32`, would use ~ 20 GiB. For Cityscapes [6], containing approximately 3000 training images of size 2048×1024 in 19 classes, this amounts to ~ 445 GiB.

Following the above observations, we introduce a relaxation of sample-wise penalties, and propose to solve the following problem:

$$
\min_{\theta} \quad \sum_{i=1}^{N} \mathcal{L}_{\text{CE}}(\boldsymbol{x}^{(i)}, y^{(i)}) + \\
\sum_{i=1}^{N} \sum_{j=1}^{K} \lambda_j \max\{0, \max_{k}\{l_k^{(i)}\} - l_j^{(i)} - m\}, \quad (5)
$$

where $(\lambda_j)_{1 \leq j \leq K} \in \mathbb{R}_+^K$. Since penalties are now classwise, we need K penalty weights. This has the advantage to scale well to denser classification tasks such as segmentation. However, we still face a challenging optimization problem (5), since we still need to chose K weights, which can be extremely complicated for large-scale datasets with many classes such as ImageNet, which contains 1000 classes. In the next section, we introduce a numerical technique to deal with this challenge.

4. Class Adaptive Network Calibration

The challenge of the previous formulation stems from correctly choosing the weights $\boldsymbol{\lambda} \in \mathbb{R}_+^K$, which can be cumbersome when K is large. Therefore, we propose to use an Augmented Lagrangian Multiplier (ALM) method to adaptively learn the weights of the penalties.

4.1. General ALM

ALM methods combine penalties and primal-dual updates to solve a constrained problem. They have well-established advantages and enjoy widespread popularity in the general context of optimization [1, 35, 41]. Specifically, we have the following generic constrained optimization problem:

$$
\min_{x} \quad f(x) \quad \text{s.t.} \quad h_i(x) \leq 0, \quad i = 1, \ldots, n \quad (6)
$$

where $f : \mathbb{R}^d \to \mathbb{R}$ is the objective function and $h_i : \mathbb{R}^d \to \mathbb{R}, i = 1, \ldots, n$ are the constraint functions. We tackle it by approximately solving a sequence $j \in \mathbb{N}$ of unconstrained problems:

$$
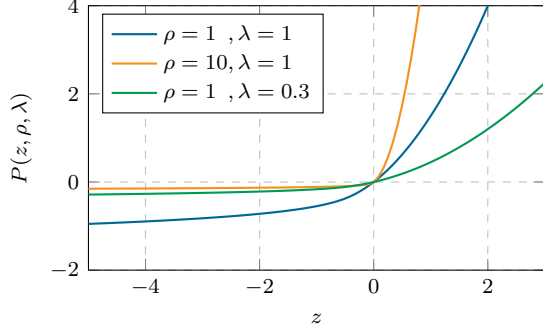\min_{x} \quad \mathcal{L}^{(j)}(x) = f(x) + \sum_{i=1}^{n} P(h_i(x), \rho_i^{(j)}, \lambda_i^{(j)}) \quad (7)
$$

Figure 2. A penalty-Lagrangian function P with varying values of ρ and μ. Higher values of ρ bring P closer to an ideal penalty. The multiplier λ is the derivative of P w.r.t. the constraint at $z = 0$.

with $P : \mathbb{R} \times \mathbb{R}_{++}^n \times \mathbb{R}_{++}^n \to \mathbb{R}$ a penalty-Lagrangian function, and $\boldsymbol{\rho}^{(j)} = (\rho_i)_{1 \leq i \leq n} \in \mathbb{R}_{++}^n$, $\boldsymbol{\lambda}^{(j)} = (\lambda_i^{(j)})_{1 \leq i \leq n} \in \mathbb{R}_{++}^n$ the penalty parameters and multipliers associated to P at the j-th iteration. This sequence of unconstrained problems is called *outer* iterations, while the steps in the minimization of $\mathcal{L}^{(j)}$ are called *inner* iterations.

The main components of ALM methods are (*i*) the *penalty-Lagrangian function* P, (*ii*) the update of the *penalty multipliers* $\boldsymbol{\lambda}^{(j)}$ and (*iii*) the increase of the *penalty parameters* $\boldsymbol{\rho}^{(j)}$. First, the penalty function P needs to satisfy a set of axioms [2] (see Appendix A): these axioms constrain the function to be continuously differentiable w.r.t. its first variable and to have a non-negative derivative: $\forall z \in \mathbb{R}, P'(z, \rho, \lambda) = \frac{\partial}{\partial z} P(z, \rho, \lambda) \geq 0$, with $P'(0, \rho, \lambda) = \lambda$. Figure 2 gives an example of a penalty, and how ρ and λ affect it. The choice of penalty function is critical to the performance of ALM methods, especially for nonconvex problems [2]. Typical functions include PHR [14,39], P_2 [17] and P_3 [34] (see section 3.2 of [2]). Second, the penalty multipliers $\boldsymbol{\lambda}^{(j)}$ are updated to the derivative of P w.r.t. the solution obtained during the last inner minimization. Formally, let $x^{(j)}$ be the approximate minimizer of $\mathcal{L}^{(j)}$, then $\forall i \in \{1, \ldots, n\}$:

$$\lambda_i^{(j+1)} = P'(h_i(x^{(j)}), \rho_i^{(j)}, \lambda_i^{(j)}) \qquad (8)$$

This update rule corresponds to a first-order multiplier estimate for the constrained problem. Third, the penalty parameters $\boldsymbol{\rho}^{(j)}$ are increased during the outer iterations if the constraints do not improve (*i.e.* is closer to being satisfied) compared to the previous outer iteration. Typically, $\rho_i^{(j+1)} = \gamma \rho_i^{(j)}$ if h_i does not improve, with $\gamma > 1$.

When the problem is convex, alternating between the approximate minimization of (7) and the update of the multipliers (8) leads to a solution for the constrained problem. The inner minimization corresponds to minimizing the primal while the outer iterations correspond to solving the dual problem. The complete procedure is presented in Algorithm 1.

Although guarantees exist only in the convex case, it is well-known that ALM methods can efficiently solve nonconvex problems as well [2]. In the context of deep learning, their use has been surprisingly underexplored [40,41].

Algorithm 1 Augmented Lagrangian Multiplier algorithm

Require: Objective function f
Require: Constraint functions $h_i, i = 1, \ldots, n$
Require: Penalty function P, initial $\boldsymbol{\lambda}^{(0)} \in \mathbb{R}_{++}^n$, $\boldsymbol{\rho}^{(0)} \in \mathbb{R}_{++}^n$
Require: Initial variable $x^{(0)}$, iterations $j = 1$
1: **while** not converged **do**
2:    Initialize with $x^{(j-1)}$ and minimize (approximately):
     $\mathcal{L}^{(j)}(x) = f(x) + \sum_{i=1}^n P(h_i(x), \rho_i^{(j)}, \lambda_i^{(j)})$
3:    $x^{(j)} \leftarrow$ (approximate) minimizer of $\mathcal{L}^{(j)}$
4:    **for** $i = 1, \ldots, n$ **do**
5:       $\lambda_i^{(j+1)} \leftarrow P'(h_i(x^{(j)}), \rho_i^{(j)}, \lambda_i^{(j)})$
6:       **if** the i-th constraint does not improve **then**
7:          $\rho_i^{(j+1)} \leftarrow \gamma \rho_i^{(j)}$
8:       **else**
9:          $\rho_i^{(j+1)} \leftarrow \rho_i^{(j)}$
10:      **end if**
11:    **end for**
12:    $j \leftarrow j + 1$
13: **end while**

4.2. ALM for calibration

Our goal now is to build an ALM method effective for calibration purposes. We can achieve this by reformulating problem (5) using a penalty function P parameterized by $(\boldsymbol{\rho}, \boldsymbol{\lambda}) \in \mathbb{R}_{++}^K \times \mathbb{R}_{++}^K$ as follows:

$$\min_{\theta} \quad \sum_{i=1}^N \mathcal{L}_{\text{CE}}(\boldsymbol{x}^{(i)}, y^{(i)}) + \sum_{k=1}^K P(d_k^{(i)} - m, \rho_k, \lambda_k) \quad (9)$$

where $d_k^{(i)} = \max\{\boldsymbol{l}^{(i)}\} - l_k^{(i)} \in \mathbb{R}_+$. With this formulation, it is natural to use a penalty-Lagrangian function for P. To avoid numerical issues typically associated with non-linear penalties, we normalize the constraints by the margin $m > 0$:

$$d_k^{(i)} - m \leq 0 \Leftrightarrow \frac{d_k^{(i)}}{m} - 1 \leq 0 \qquad (10)$$

This leads to improved numerical stability for the ALM multiplier update as well. Additionally, we average the constraints instead of summing them. This makes the method independent of the number of classes, and eases the choice of initial penalty parameters $\boldsymbol{\rho}^{(0)}$. The resulting loss is:

$$\sum_{i=1}^N \mathcal{L}_{\text{CE}}(\boldsymbol{x}^{(i)}, y^{(i)}) + \frac{1}{K} \sum_{k=1}^K P\left(\frac{d_k^{(i)}}{m} - 1, \rho_k, \lambda_k\right) \quad (11)$$

As noted in Section 3, one of the main downsides of estimating Lagrange multipliers from the training set is that

we could quickly overfit the data. Therefore, we propose to use the validation set to obtain a reliable estimate of the penalty multipliers at each epoch. We consider that an epoch of training corresponds to the approximate minimization of the loss function, and then compute the average penalty multiplier on the validation set. Formally after a training epoch j, the penalty multipliers for epoch $j+1$ will be, for all $k = 1, \ldots, K$:

$$\lambda_k^{(j+1)} = \frac{1}{|\mathcal{D}_{\text{val}}|} \sum_{(\boldsymbol{x}, y) \in \mathcal{D}_{\text{val}}} P'\left(\frac{d_k}{m} - 1, \rho_k^{(j)}, \lambda_k^{(j)}\right) \quad (12)$$

Finally, the penalty multiplier is projected on a safeguarding interval $[\lambda_{\min}, \lambda_{\max}] = [10^{-6}, 10^6]$ in our case. To update the penalty parameters $\boldsymbol{\rho}$, we compute the average constraint per class on the validation set. Then, for each class, if the average constraint is positive and has not decreased compared to the previous epoch, we multiply the corresponding penalty parameter by γ.

Finally, as suggested by [2] and confirmed by our empirical results, we utilize the PHR function in our implementation, defined as follows:

$$\text{PHR}(z, \rho, \lambda) = \begin{cases} \lambda z + \frac{1}{2}\rho z^2 & \text{if} \quad \lambda + \rho z \geq 0; \\ -\frac{\lambda^2}{2\rho} & \text{otherwise.} \end{cases} \quad (13)$$

Overall, the proposed method, consolidated in Algorithm 2, corresponds to approximately solving the constrained problem (2), by learning class-wise penalty multipliers on the validation set, to avoid overfitting and training specificities (*i.e.* batch normalization, dropout, augmentations) which would result in unreliable penalty multipliers estimates.

5. Experiments

5.1. Experimental Setup

Datasets. We perform experiments on a variety of popular benchmarks. First, we include three widely used image classification benchmarks, including *Tiny-ImageNet* [7], *ImageNet* [7] and one long-tailed image classification, *ImageNet-LT* [25]. Tiny-ImageNet is widely used in the calibration literature [22, 31], with relatively small 64×64 resolution, while ImageNet [7] is a large-scale benchmark consisting of 1000 categories and over 1M images. The main characteristic of ImageNet-LT is that the number of samples is extremely imbalanced across classes, ranging from 5 to 1280. To evaluate performance in dense prediction tasks, we include one semantic segmentation benchmark, *PASCAL VOC2012* [9]. Furthermore, one benchmark from the NLP domain, *20 Newsgroups* [19], is included to show the general applicability. For a detailed description of each dataset and the pre-processing settings, please refer to Appendix B.

Evaluation Metrics. For calibration, we report the most widely used Expected Calibration Error (ECE) [33]. Samples

---

Algorithm 2 CALS-ALM training

Require: DNN initial θ^0, margin m
Require: Dataset: $\mathcal{D}_{\text{train}}, \mathcal{D}_{\text{val}}$, batch size B
Require: Penalty function $P, \gamma > 1, \tau \in (0, 1)$
Require: Initial $\boldsymbol{\lambda}^{(0)} \in \mathbb{R}_{++}^n, \boldsymbol{\rho}^{(0)} \in \mathbb{R}_{++}^n$
1: **for** $j = 0, \ldots, T$ **do**   // Epochs of training
2:    **for** each mini-batch $\{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^B$ in $\mathcal{D}_{\text{train}}$ **do**
3:      $\mathcal{L}_{\text{c}} = \sum_{i=1}^B \mathcal{L}_{\text{CE}}(\boldsymbol{x}^{(i)}, y^{(i)})$ // Cross-entropy
4:      $\mathcal{L}_{\text{p}} = \sum_{i=1}^B \frac{1}{K} \sum_{k=1}^K P\left(\frac{d_k^{(i)}}{m} - 1, \rho_k^{(j)}, \lambda_k^{(j)}\right)$ // Penalties
5:      $\mathcal{L} = \frac{1}{B}(\mathcal{L}_{\text{c}} + \mathcal{L}_{\text{p}})$
6:      $\theta^{(t+1)} \leftarrow \theta^{(t)} - \alpha \cdot \nabla_\theta \mathcal{L}$ // Gradient descent
7:    **end for**
8:    **for** $k = 1, \ldots, K$ **do**   // Adjust λ and ρ on validation
9:      $\lambda_k^{(j+1)} = \frac{1}{|\mathcal{D}_{\text{val}}|} \sum_{(\boldsymbol{x}, y) \in \mathcal{D}_{\text{val}}} P'\left(\frac{d_k}{m} - 1, \rho_k^{(j)}, \lambda_k^{(j)}\right)$
10:      $\overline{d_k}^{(j)} = \frac{1}{|\mathcal{D}_{\text{val}}|} \sum_{(\boldsymbol{x}, y) \in \mathcal{D}_{\text{val}}} \frac{d_k}{m} - 1$ // Average constraint
11:      **if** $j \geq 1$ **and** $\overline{d_k}^{(j)} > \tau \max\{0, \overline{d_k}^{(j-1)}\}$ **then**
12:        $\rho_k^{(j+1)} \leftarrow \gamma \rho_k^{(j)}$ // Constraint has not improved
13:      **else**
14:        $\rho_k^{(j+1)} \leftarrow \rho_k^{(j)}$
15:      **end if**
16:    **end for**
17: **end for**

---

are grouped into M equi-spaced bins according to prediction confidence, and a weighted average of the absolute difference between accuracy and confidence in each bin is calculated:

$$ECE = \sum_{m=1}^M \frac{|B_m|}{N} |A_m - C_m|, \quad (14)$$

where M is the number of bins, N the amount of test samples, B_m the samples with prediction confidence in the mth bin, A_m the accuracy and C_m the mean confidence of samples in the mth bin. Note we fix N to 15 according to [22, 31]. In accordance with [22], we also report Adaptive ECE (AECE), a variant of ECE where the bins are configured to evenly distribute the test samples across them. Additionally, Classwise Calibration Error (CWCE) [29], a classwise extension of ECE, is included in Appendix C. For discriminative performance, we use standard measures: accuracy (Acc) for classification, and intersection over union (mIoU) for segmentation.

Compared methods. We compare our method to other learning based calibration losses, including *(i)* methods that impose constraints on predictions (either logits or softmax probabilities), *i.e.* Explicit Confidence Penalty (ECP) [38], Label Smoothing (LS) [42], Focal Loss (FL) [21] and its sample-dependent version (FLSD) [31], Margin-based Label

Method	TinyImageNet ResNet-50			ImageNet ResNet-50			ImageNet SwinV2-T			ImageNet-LT ResNet-50			ImageNet-LT SwinV2-T		
	Acc	ECE	AECE	Acc	ECE	AECE	Acc	ECE	AECE	Acc	ECE	AECE	Acc	ECE	AECE
CE	65.02	3.73	3.69	75.16	9.19	9.18	75.60	9.95	9.94	37.90	28.12	28.12	31.82	31.82	36.68
MMCE [18]	65.34	2.81	2.61	74.85	8.57	8.56	76.68	9.07	9.08	37.79	28.41	28.40	33.14	26.41	26.41
ECP [38]	64.90	4.00	3.92	75.22	8.27	8.26	75.82	9.88	9.86	37.69	28.14	28.13	31.22	33.70	33.70
LS [42]	65.78	3.17	3.16	76.04	2.57	2.88	75.42	7.32	7.33	37.88	10.46	10.38	31.70	11.42	11.40
FL [31]	63.09	2.96	3.12	73.87	1.60	1.65	75.60	3.19	3.18	36.04	18.37	18.36	30.73	25.50	25.50
FLSD [31]	64.09	2.91	2.95	73.97	2.08	2.06	74.70	2.44	2.37	36.18	17.77	17.78	32.56	25.16	25.17
CPC [5]	64.49	4.88	4.91	76.33	3.66	3.59	76.34	5.50	5.33	38.90	16.00	15.99	32.54	13.21	13.19
MbLS [22]	64.74	1.64	1.73	75.82	4.44	4.26	77.18	1.95	1.73	38.32	6.16	6.16	32.05	7.65	7.64
CALS-HR	65.09	2.50	2.42	76.34	5.63	5.69	77.58	3.06	2.95	38.50	2.83	2.78	34.31	2.37	2.45
CALS-ALM	65.03	1.54	1.38	76.44	1.46	1.32	77.10	1.61	1.69	38.56	2.15	2.30	33.94	2.32	2.45

Table 1. Calibration performance for different approaches on three image classification benchmarks. We report two lower-is-better calibration metrics, *i.e.* ECE and AECE. Best method is highlighted in bold, while the second-best one is underlined.

Smoothing (MbLS) [22] and CPC [5], and *(ii)* techniques that directly optimize calibration measures, *i.e.* MMCE [18]. We refer to the related literature [22, 31] to set the hyper-parameters for various methods. For instance, the smoothing factor in LS and FL is set to 0.05 and 3 respectively, and we set margin to 10 in MbLS. A detailed description of hyper-parameter values can be found in Appendix F.

Our methods. A simple alternative to the algorithm presented in Section 4.2 would be to heuristically tune multipliers by scaling them according to penalty values: if $P_k^{(j+1)}$ increases we also increase λ_k^{j+1} and vice versa. This strategy, akin to learning rate scheduling, can be formulated as:

$$\lambda_k^{(j+1)} = \begin{cases} \mu \lambda_k^{(j)} & \text{if} \quad P_k^{(j+1)} > \tau P_k^{(j)} \\ \lambda_k^{(j)}/\mu & \text{if} \quad P_k^{(j)} > \tau P_k^{(j+1)} \\ \lambda_k^{(j)} & \text{otherwise} \end{cases} \quad (15)$$

where $\mu > 1$ and $\tau > 1$ are hyper-parameters that we fix to 1.1. We refer to our main algorithm as **CALS-ALM** and to this heuristic rule as **CALS-HR** in what follows.

We fix the margin to $m = 10$ on vision tasks and $m = 6$ on the NLP benchmark, as in [22], for a fair comparison. We also perform an ablation study to investigate the impact of the margin value. For other hyper-parameters, we set $\boldsymbol{\lambda}^{(0)} = 10^{-6} \cdot \mathbf{1}_K$, $\boldsymbol{\rho}^{(0)} = \mathbf{1}_K$, $\gamma = 1.2$, and we update the penalty parameters $\boldsymbol{\rho}$ every 10 epochs. Please refer to Appendix F for a detailed description of all hyper-parameters.

Implementation Details. For image classification, we experiment with ResNet [13] and a vision Transformer model, *i.e.* Swin Transformer V2 (SwinV2-T) [23]. DeepLabV3 [3] is employed for semantic segmentation on PASCAL VOC2012. Following [22, 31], we use the Global Pooling CNN (GPool-CNN) architecture [20] on the NLP recognition task. Further training details on each dataset can be found in Appendix B.

5.2. Results

Results on image classification. Table 1 presents the discriminative and calibration performance of our methods across three widely used classification benchmarks, compared to baselines and related works. We can observe that our CALS-ALM approach consistently outperforms existing techniques in terms of calibration. Specifically, the results indicate that the standard CE loss and other approaches often lead to miscalibrated models, with the severity of miscalibration substantially increasing in correlation with dataset difficulty. This is particularly evident in large-scaled datasets with numerous classes, such as ImageNet, or those with long-tail class distributions, like ImageNet-LT. Although other learning-based methods could provide better calibrated networks, their performance is not stable across different settings. For example, while FL achieves a relatively low ECE of 1.60 with a Resnet50 trained on ImageNet, it only yields an ECE of 25.50 when training a SwinV2-T network on ImageNet-LT, revealing a limitation in adapting to different learning scenarios. In contrast, CALS-ALM attains the best calibration performance in all cases, frequently outperforming exsiting approaches by a significant margin, with minimal variations across datasets and architectures. This trends persists when compared to the most closely related technique, MbLS. It is noteworthy that the improvements on the long-tailed ImageNet-LT dataset are substantial; for example, we decrese ECE from 28.12 to 2.15 for ResNet-50, and from 31.82 to 2.32 for SwinV2-T, validating the effectiveness of class adaptive learning. Another interesting finding is that CALS-HR, employing a naive update strategy for class-wise penalty weights, achieves nearly the second-best performance in the ImageNet-LT dataset. This further demonstrates the effectiveness of CALS for learning under class-imbalance scenarios. For the reliability diagrams of various models, please refer to Appendix E.

In terms of model accuracy, our method delivers competitive performances, surpassing existing methods in certain cases. It is important to emphasize that, while the proposed method achieves discriminative results comparable to the best-performing approach for each dataset, the differences in calibration are considerable. This highlights the superiority of the proposed formulation for training highly discrimi-
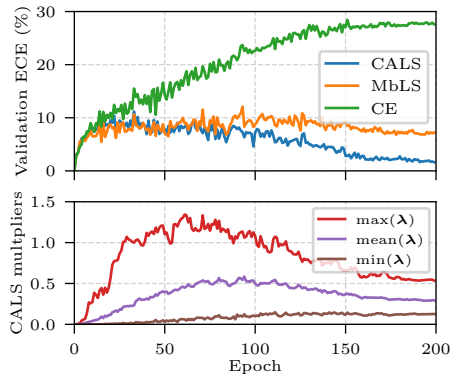
(a) Validation ECE and multipliers during training.
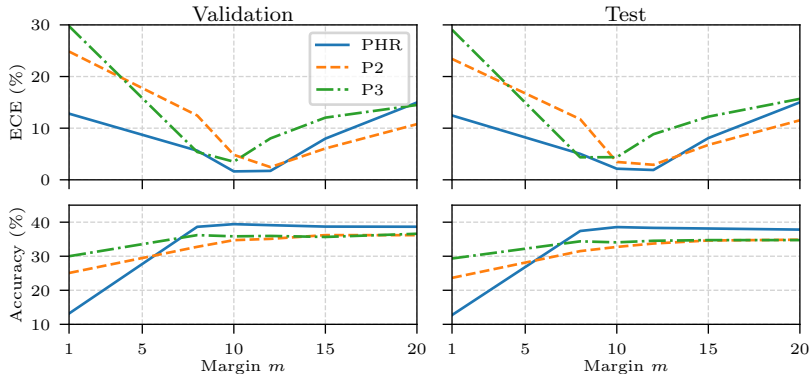
(b) Effect of penalty function and margin on accuracy and ECE for CALS.

Figure 3. **Ablation study on ImageNet-LT.** (a) **Evolution of ECE on validation and multipliers for CALS**: ECE on the validation set for our method (CALS), CE and MbLS [22] and values of multipliers $\boldsymbol{\lambda}$ for CALS after each training epoch. (b) **Effect of penalty functions and margin**: ECE and accuracy on validation and test set are shown across different choices of penalty functions and margin values.

native and well-calibrated networks. Figure 1 provides a more visual comparison considering both accuracy and ECE. It is demonstrated that CALS-ALM provides the optimal compromise between accuracy and calibration performance.

Ablation Analysis. Figure 3 illustrates the evolution of ECE in (a) and penalty multipliers $\boldsymbol{\lambda}$ in (b) during training. It is interesting to observe that the evolution of $\boldsymbol{\lambda}$ is consistent with the ECE. Specifically, the average penalty weight gradually increases while the ECE initially deteriorates because the model is focused on increasing accuracy. However, the value of the penalty weight begins to decline alongside the ECE, as the network starts to become better calibrated. For a visualization of classwise multipliers, please refer to Appendix D. Figure 3(c) highlights the impact of the choice of penalty functions and margin values. This demonstrates that the PHR penalty function is preferable over the other two options, P2 and P3, for both calibration and accuracy. Regarding the margin, the best performance is achieved with $m \approx 10$, which is consistent with the findings in [22].

Semantic Segmentation. Table 2 presents the performance on Pascal VOC dataset. Note here CALS refers to our best method, *i.e.* CALS-ALM. It can be observed that the trend is consistent with image classification experiments: CALS outperforms counterparts in terms of ECE, and yields competitive results on discriminative performance, *i.e.* mIoU in segmentation. It is worth noting that some methods like MMCE, FLSD and CPC, are not included here because their computation demands were too heavy for pixel-wise segmentation tasks. In contrast, our method is unlimited in dense prediction tasks as the computation cost it adds is moderate.

Text Classification. Last, we demonstrate the general applicability of the proposed method by analyzing its performance on a non-vision task, *i.e.* text classification on 20 Newsgroups dataset. The results are reported in Table 3. Remarkably, CALS again brings substantial improvement in

	CE	ECP	LS	FL	MbLS	CALS
ECE	14.75	<u>5.15</u>	6.90	10.87	5.22	**4.66**
mIoU	66.46	65.57	**67.73**	64.25	65.29	<u>66.77</u>

Table 2. Segmentation results on the PASCAL VOC 2012.

	CE	MMCE	ECP	LS	FL	FLSD	CPC	MbLS	CALS
ECE (%)	22.75	23.02	22.97	8.07	10.80	10.87	9.46	<u>5.40</u>	**2.04**
Acc (%)	67.01	66.23	66.48	67.14	66.08	65.85	<u>68.27</u>	67.89	**68.32**

Table 3. Results on the text classification task, 20 Newsgroups.

terms of calibration, with ECE decreasing to 2.04%, while yielding the best accuracy 68.32%. This reveals that the proposed class adaptive learning method is also able to handle class differences in NLP applications and provide promising performance in terms of both accuracy and calibration.

6. Limitations and Future Work

We have proposed Class Adaptive Label Smoothing for network calibration based on a modified Augmented Lagrangian Multiplier algorithm. Despite its superior performance over previous methods, there are potential limitations in this work. For instance, our method requires the validation set to have the same distribution as the training set. Although this is satisfied in nearly every benchmark, it will be interesting to investigate the impact of using non-independent and identically distributed (*i.i.d*) validation sets.

Acknowledgements

References

[1] Dimitri P. Bertsekas. *Constrained Optimization and Lagrange Multiplier Methods (Optimization and Neural Computation Series)*. Athena Scientific, 1 edition, 1996. 4

[2] Ernesto G Birgin, Romulo A Castillo, and José Mario Martínez. Numerical comparison of augmented lagrangian algorithms for nonconvex problems. *Computational Optimization and Applications*, 31(1), 2005. 5, 6, 11

[3] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. In *CVPR*, 2017. 1, 7, 11

[4] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, 2020. 1

[5] Jiacheng Cheng and Nuno Vasconcelos. Calibrating deep neural networks by pairwise constraints. In *CVPR*, 2022. 3, 7, 13

[6] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, June 2016. 4

[7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 6, 11

[8] Zhipeng Ding, Xu Han, Peirong Liu, and Marc Niethammer. Local temperature scaling for probability calibration. In *ICCV*, 2021. 1, 3

[9] Mark Everingham, S. M. Eslami, Luc Gool, Christopher K. Williams, John Winn, and Andrew Zisserman. The pascal visual object classes challenge: A retrospective. *IJCV*, 111(1):98–136, 2015. 4, 6, 11

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

[11] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *ICML*, 2017. 1, 2, 3, 12

[12] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *CVPR*, 2022. 1

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1, 7, 11

[14] Magnus R. Hestenes. Multiplier and gradient methods. *Journal of Optimization Theory and Applications*, 4:303–320, 1969. 5

[15] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 1, 2

[16] Archit Karandikar, Nicholas Cain, Dustin Tran, Balaji Lakshminarayanan, Jonathon Shlens, Michael C Mozer, and Rebecca Roelofs. Soft calibration objectives for neural networks. In *NeurIPS*, 2021. 3

[17] Barry W. Kort and Dimitri P. Bertsekas. Combined primal–dual and penalty methods for convex programming. *SIAM Journal on Control and Optimization*, 14(2):268–294, 1976. 5

[18] Aviral Kumar, Sunita Sarawagi, and Ujjwal Jain. Trainable calibration measures for neural networks from kernel mean embeddings. In *ICML*, 2018. 3, 7, 13

[19] Ken Lang. Newsweeder: Learning to filter netnews. In *ICML*, 1995. 6, 12

[20] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. In *ICLR*, 2014. 7, 12

[21] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *CVPR*, 2017. 1, 3, 6

[22] Bingyuan Liu, Ismail Ben Ayed, Adrian Galdran, and Jose Dolz. The devil is in the margin: Margin-based label smoothing for network calibration. In *CVPR*, 2022. 1, 2, 3, 6, 7, 8, 11, 12, 13

[23] Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, Furu Wei, and Baining Guo. Swin transformer v2: Scaling up capacity and resolution. In *CVPR*, 2022. 7, 11

[24] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*, 2021. 1

[25] Ziwei Liu, Zhongqi Miao, Xiaohang Zhan, Jiayun Wang, Boqing Gong, and Stella X. Yu. Large-scale long-tailed recognition in an open world. In *CVPR*, 2019. 6, 11

[26] Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. In *ICLR*, 2017. 11

[27] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019. 1, 11

[28] Xingchen Ma and Matthew B. Blaschko. Meta-cal: Well-controlled post-hoc calibration by ranking. In *ICML*, 2021. 3

[29] Lena Maier-Hein et al. Metrics reloaded: Pitfalls and recommendations for image analysis validation. *arXiv preprint arXiv:2206.01653*, 2022. 6, 12

[30] Matthias Minderer, Josip Djolonga, Rob Romijnders, Frances Hubis, Xiaohua Zhai, Neil Houlsby, Dustin Tran, and Mario Lucic. Revisiting the calibration of modern neural networks. In *NeurIPS*, 2021. 3, 11, 12

[31] Jishnu Mukhoti, Viveka Kulharia, Amartya Sanyal, Stuart Golodetz, Philip HS Torr, and Puneet K Dokania. Calibrating deep neural networks using focal loss. In *NeurIPS*, 2020. 1, 2, 3, 6, 7, 11, 12, 13

[32] Rafael Müller, Simon Kornblith, and Geoffrey Hinton. When does label smoothing help? In *NeurIPS*, 2019. 1, 3

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

[34] H. Nakayama, H. Samaya, and H. Samaya. A generalized lagrangian function and multiplier method. *Journal of Optimization Theory and Applications*, 17:211–227, 1975. 5

[35] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer, New York, NY, USA, 2 edition, 2006. 4

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

[37] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *EMNLP*, 2014. 12

[38] Gabriel Pereyra, George Tucker, Jan Chorowski, Łukasz Kaiser, and Geoffrey Hinton. Regularizing neural networks by penalizing confident output distributions. In *ICLR*, 2017. 1, 3, 6, 7, 13

[39] M. J. D. Powell. A method for nonlinear constraints in minimization problems. In *Fletcher, R., Ed., Optimization, Academic Press, New York, NY*, 1969. 5

[40] Jérôme Rony, Eric Granger, Marco Pedersoli, and Ismail Ben Ayed. Augmented lagrangian adversarial attacks. In *ICCV*, 2021. 5

[41] Sara Sangalli, Ertunc Erdil, Andeas Hötker, Olivio F Donati, and Ender Konukoglu. Constrained optimization to train neural networks on critical and under-represented classes. In *NeurIPS*, 2021. 4, 5

[42] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, 2016. 3, 6, 7, 13

[43] Christian Tomani, Sebastian Gruber, Muhammed Ebrar Erdem, Daniel Cremers, and Florian Buettner. Post-hoc uncertainty calibration for domain drift scenarios. In *CVPR*, 2021. 3

[44] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. 1