Enhancing Confidence Calibration in Long-Tailed Recognition

A dissertation submitted in
partial fulfilment for the degree of

Master of Technology

in

Computer Science

by

Sasanka Jana
Roll no. - [**CS2227**]

under the supervision of

Dr. Swagatam Das
Electronics and Communication Sciences Unit (ECSU)

Indian Statistical Institute, Kolkata
June, 2024

CERTIFICATE

This is to certify that the thesis titled, **"Enhancing Confidence Calibration in Long-Tailed Recognition"**, authored by **Mr. Sasanka Jana**, Roll Number **CS2227**, is an original piece of work conducted under my supervision. To the best of my knowledge, neither this thesis nor any part of it has been previously submitted for any academic degree, diploma, or title.

I believe this thesis meets all the criteria for the award of the degree of **Master of Technology in Computer Science**.

June,2024

Dr. Swagatam Das
Professor
Electronics and Communication Sciences Unit,
Indian Statistical Institute,
Kolkata-700108

Acknowledgement

I wish to extend my sincere thanks to my advisor, Dr. Swagatam Das, for the opportunity to complete this dissertation. I am deeply grateful to Mr. Faizan Ansari, SRF, whose assistance and expertise in deep learning techniques and PyTorch coding were invaluable during my research.

I also appreciate the computational resources provided, which were essential for my research. Additionally, I am thankful to my friends for their support, encouragement, and companionship.

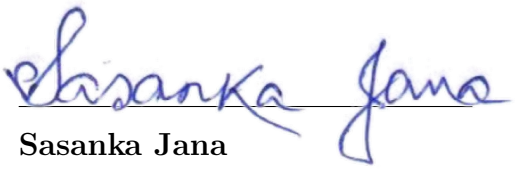
Declaration of Originality

I, **Sasanka Jana**, with Roll No. **CS2227**, hereby declare that the content presented in the dissertation titled

Enhancing Confidence Calibration in Long-Tailed Recognition

is my original work, conducted for the degree of **Master of Technology in Computer Science** at the **Indian Statistical Institute, Kolkata**.
By signing this declaration, I certify that:

- None of the data or results have been manipulated.
- No plagiarism or intellectual property theft has been committed.
- All contributions of others have been properly acknowledged and cited.
- I understand that this work may be subjected to screening for academic misconduct.



Sasanka Jana
Roll No. CS2227

Abstract

Deep neural networks often struggle with heavily class-imbalanced training datasets. Recently, two-stage methods have been developed to separate representation learning from classifier learning, aiming to enhance performance. However, the crucial issue of **miscalibration** remains. To tackle this, we introduce novel methods to improve both calibration and performance in such scenarios.

Recognizing that predicted probability distributions of classes are closely tied to the number of class instances, we propose label-aware smoothing with balanced softmax. This strategy tackles the issue of differing levels of over-confidence among different categories, thereby improving the learning process of classifiers. Furthermore, to counteract potential bias in the dataset between the two stages caused by different sampling techniques, we incorporate shifted batch normalization into the decoupling framework.

The methods we suggest have set fresh standards on numerous prevalent long-tailed recognition datasets such as CIFAR10-LT and CIFAR100-LT.

Keywords: Classification, class imbalance, Balance Softmax, miscalibration.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Calibration [1] in Classification

Deep neural network classifiers have been shown to be mis-calibrated for long-tailed recognition i.e., their prediction probabilities are not reliable confidence estimates. For example, if a neural network classifies an image as a "cat" with probability p, p cannot be interpreted as the confidence of the network's predicted class for the image. Further, neural network classifiers are often overconfident [15] in their predictions. A calibrated neural network classifier produces class probabilities that accurately reflect the actual likelihood of the class being correct in the ground truth dataset However, a critical issue remains – **miscalibration** [15]. In addressing this obstacle, we present novel techniques aimed at enhancing calibration and efficacy within these circumstances.

Let's consider an example of a deep learning model employed for multi-class classification, such as a model that predicts the type of flower in an image. Suppose we have a dataset of 2000 flower images, each labeled with one of three possible categories: daisy, rose, or sunflower. We train a deep learning model to classify these images into one of these three categories.

After training, we assess the model's performance on a separate test set of 400 images. The model produces a probability score for each category, indicating the likelihood that the image belongs to that category. For example, the model might output a score of 0.7 for the rose category, 0.15 for the daisy category, and 0.15 for the sunflower category for a particular image.

However, these probability scores might not precisely represent the true likelihood of each category. For example, if the model assigns an 70% probability to an image being a rose, but only 60 out of 100 images labeled as roses truly contain roses, then our model needs calibration.

Calibration [1] in deep learning models involves adjusting the model's predicted

probabilities to make them align more closely with the actual likelihood of the predicted categories. For example, we can use isotonic regression to calibrate the model's predicted probabilities. After calibration, the model might output a score of 0.6 for the rose category, 0.2 for the daisy category, and 0.2 for the sunflower category for the same image.

By employing calibration methods, we can refine our model's predictions to more closely match the real outcomes, resulting in greater accuracy and reliability. Once calibrated, we can trust that the predicted probabilities truly represent the actual likelihood of each category. These probabilities can then be utilized for decision-making tasks, such as determining the type of flower in an image.

1.2 Accuracy

Accuracy refers to the degree of correctness or precision of a measurement or prediction. In various contexts, accuracy can have different meanings, but generally, it assesses how close a value or result is to the true value. The accurecy is defined as:

$$\text{Accuracy} = \frac{\text{Correct predictions}}{\text{All predictions}}$$

In the context of model evaluation, accuracy and calibration serve distinct but complementary purposes. **Accuracy** measures the correctness of predictions by comparing predicted labels to true labels, providing a straightforward metric of overall performance. For instance, a model with 95% accuracy correctly classifies 95 out of 100 instances. **Calibration** [1], on the other hand, assesses the reliability of the predicted probabilities, ensuring that the confidence levels reflect the true likelihood of outcomes. This is crucial for applications where probabilistic predictions guide decision-making, such as in medical diagnoses or risk assessment. While accuracy indicates how often a model is right, calibration ensures that the predicted probabilities are trustworthy, both of which are essential for developing robust and reliable models.

1.3 Expected Calibration Error (ECE)

Model calibration aims to align a model's predictions with the true probabilities, ensuring that the predictions are both reliable and accurate. While there are various methods to measure calibration, this article focuses on explaining and demonstrating one straightforward yet effective metric: Expected Calibration Error (ECE).

ECE calculates the weighted average error of the estimated probabilities, providing a single value that allows for the comparison of different models. This makes it a practical and sufficient measure for assessing model calibration.

Definition: The ECE evaluates the alignment between a model's predicted probabilities and the actual probabilities by computing a weighted mean of the absolute disparity between accuracy (acc) and confidence (conf):

$$ECE = \frac{1}{n}\sum_{m=1}^{M} |B_m| \, |acc(B_m) - conf(B_m)|$$

The measure involves splitting the data into M equally spaced bins. B is used for representing "bins" and m for the bin number. We'll get back to the individual parts of this formula such as B, $|B_m|$, acc(B_m), and conf(B_m) in more detail later.

conf(B_m) represents the average estimated probabilities in bin m, defined as:

$$\text{confidence of } (B_m) = \sum_{i \in B_m} \frac{\hat{p}_i}{|B_m|}$$

acc(B_m) represents the accuracy per bin m, defined as:

$$\text{accurecy of } (B_m) = \sum_{i \in B_m} \frac{\mathbf{1}(\hat{y}_i = y_i)}{|B_m|}$$

1.4 Real life data and Class Imbalance

In many real-world scenarios, data often exhibit a class imbalance, where a few instances significantly outweigh the others. This phenomenon is particularly prevalent in various fields, leading to challenges in developing effective machine learning models. Class imbalance arises when there is an uneven distribution of classes, resulting in one class (the minority class) having notably fewer instances compared to the other class (the majority class).

Examples of Class Imbalance:

In the iNaturalist 2017 (iNat2017) [12] dataset, there exists a significant class imbalance. This dataset comprises 859,000 images representing over 5,000 different species of plants and animals. The imbalance ratio (IR) of 435.44 indicates a highly skewed distribution of images across species.

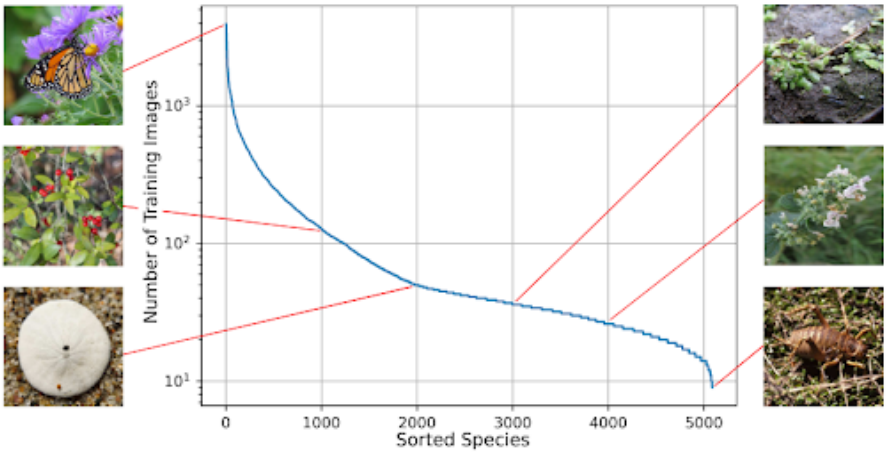


Figure 1.1: Training picture distribution by species. There is a significant imbalance between the classes in iNat2017 [12], with almost 16% of training photos found in the top 1% most populous courses.

Figure 1.2: Examples of images from the Places Database categorized into two sets per category [13]. Three macro-classes—Indoor, Nature, and Urban—are present in the dataset.

The Places Database [13] is a comprehensive collection of 10 million scene photographs, meticulously labeled with scene semantic categories. This dataset is designed to encompass a vast array of environments, reflecting the diversity and richness of the types of scenes encountered globally.

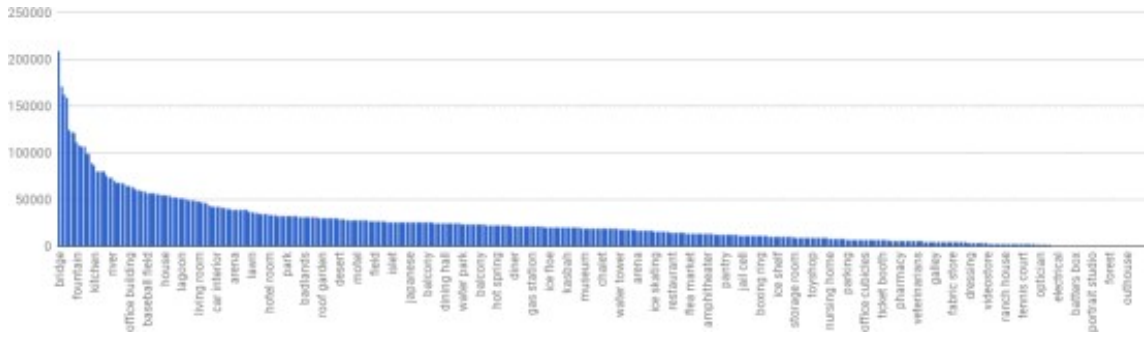


Figure 1.3: Places Database picture number distribution sorted by category [13]. 10,624,928 photos from 434 categories are in the Places collection. Every six intervals, the category names are displayed.

1.5 Problem Definition

In the realm of deep neural networks, training on datasets with substantial class imbalances poses a notable challenge. Recently, there has been a trend toward adopting two-stage methodologies, separating the tasks of representation learning and classifier learning to boost performance. Nevertheless, a critical issue that persists is **miscalibration** [15].

To address this concern, we've introduced methods specifically crafted to enhance calibration and elevate overall performance in these challenging scenarios.

Chapter 2

Review of related work

Convolutional Neural Networks (CNNs) often struggle with class imbalance due to several mathematical reasons:

- **Limited Samples of Minority Classes**: Modern CNNs typically have millions of parameters that require diverse examples for effective generalization. When there are few examples of minority classes, the model struggles to learn robust and discriminative features, leading to poor classification accuracy for those classes.

- **Biased Gradients**: CNNs are trained by minimizing a loss function, such as cross-entropy loss. In imbalanced datasets, where one class is much more prevalent than others, the gradients of the loss function during training become biased towards the majority class.Each training batch mainly consists of majority class data points, leading the model to prioritize minimizing errors for the majority class while neglecting the minority class.

Consider a CNN trained to detect various medical conditions from X-ray images. The dataset may include thousands of images for common conditions like pneumonia but only a few for rare conditions like tuberculosis or certain types of tumors. The scarcity of images for rare conditions hampers the model's ability to learn distinctive features of these conditions. As a result, the CNN might not perform well in identifying rare diseases, leading to high false-negative rates. During training, the majority of X-ray images belong to common conditions. This leads to gradients that are biased towards minimizing errors for common conditions. Consequently, the model may become highly accurate in detecting common conditions but fails to detect rare diseases accurately. This imbalance in training data means the CNN might not learn effective decision boundaries for rare conditions. In the high-dimensional space where X-ray images are represented, the decision boundaries for rare conditions might be poorly defined or overly simplistic, leading to misclassification.

To address these challenges, various techniques have been proposed. These include data augmentation to artificially increase the number of minority class samples, resampling methods such as oversampling the minority class or undersampling the majority class, cost-sensitive learning approaches like focal loss or weighted cross-entropy, and specialized two-stage learning methods. These techniques aim to rebalance the training process, amplify the impact of minority samples, or modify the loss function to better handle class imbalance issues.

2.1 Re-sampling

Re-sampling techniques are commonly used to address the issue of class imbalance in datasets. Below different types of re-sampling methods

a. **Random Under-Sampling [16] :** In contrast to over-sampling, this method randomly removes samples from the majority class to reduce its dominance in the dataset. This technique aims to balance the class distribution by reducing the number of instances of the majority class.

b. **Random Over-Sampling [17] :**This method entails randomly duplicating samples from the minority class to enhance its representation in the dataset. By doing so, the model can have more instances of the minority class to learn from, which helps in balancing the class distribution.

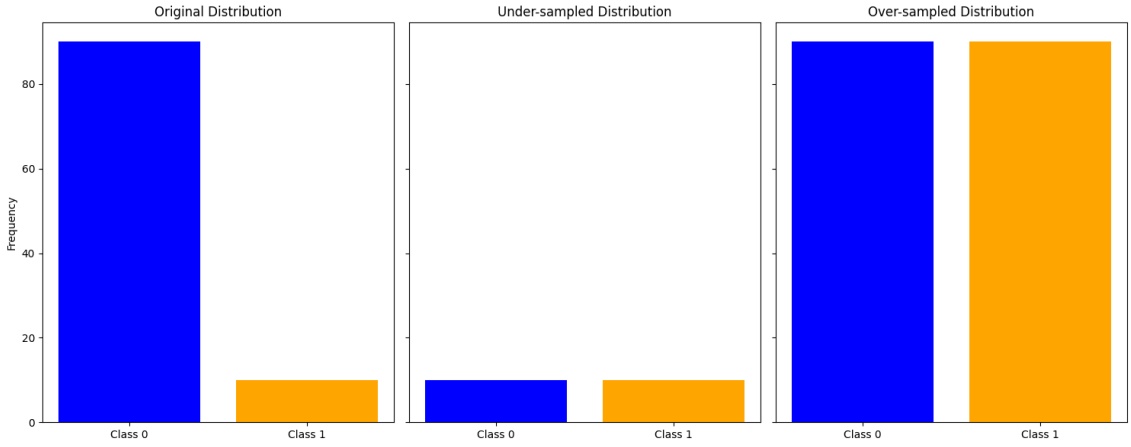


Figure 2.1: Undersampling vs oversampling on a dataset.

c. **SMOTE (Synthetic Minority Over-sampling Technique) [18]:** SMOTE[18] generates synthetic samples for the minority class by interpolating between existing minority class samples. This method helps in creating new, plausible samples that are not mere duplicates, thus enriching the minority class with more diverse examples.

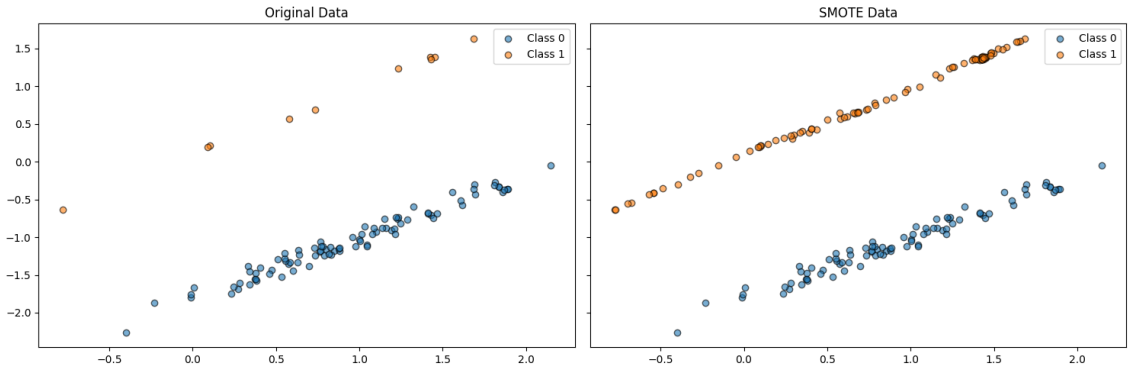


Figure 2.2: SMOTE

d. **Cluster-Based Over-Sampling [19]:** This method involves clustering the majority class samples and then over-sampling the minority class within each cluster. By ensuring that synthetic samples are created in a way that respects the structure of the data, this technique helps in maintaining the natural distribution and reducing the risk of overfitting.

2.2 Re-weighting

Re-weighting techniques equalize the class distribution by modifying the weights assigned to training instances within the loss function. This involves assigning higher weights to minority classes and lower weights to majority classes.

2.2.1 Weighted Cross Entropy [2]

Weighted Cross Entropy [2], introduces a scaling factor α to Binary Cross Entropy, enabling more stringent penalties for false positives or false negatives. If you want false positives to be penalized more than false negatives, α must be greater than 1. Otherwise, it must be less than 1.
The equations for Binary Cross Entropy and Weighted Cross Entropy [2] Loss are the following:

$$L_{\text{BCE}} = -y \log(\hat{y}(z)) - (1-y) \log(1-\hat{y}(z)) \ ,$$
$$L_{\text{WCE}} = -\alpha y \log(\hat{y}(z)) - (1-y) \log(1-\hat{y}(z))$$

2.2.2 Focal loss[3]

Focal loss[3] is designed to address the problem of class imbalance by down-weighting the contribution of easy-to-classify examples, thereby focusing more on hard examples. This is particularly useful in scenarios where there are vast amounts of easily

classified negative samples that can overwhelm the training process.

Formulation

Let $y \in \{\pm 1\}$ be the ground-truth class, and let p be the estimated probability for the class where $y = 1$. The posterior probability p_t is defined as:

$$p_t = \begin{cases} p, & \text{if } y = 1 \\ 1 - p, & \text{if } y = -1 \end{cases},$$

where $p = \text{sigmoid}(x)$.

The Binary Cross-Entropy (BCE) Loss is given by:

$$\mathcal{L}_{\text{BCE}}(p_t) = -\log(p_t).$$

The gradient of the BCE Loss is:

$$\frac{d\mathcal{L}_{\text{BCE}}(p_t)}{dx} = y(p_t - 1)$$

In the case of class imbalance, the gradients can be dominated by easily classified samples, particularly negative ones. To mitigate this, Focal loss[3] introduces a modulating factor $(1 - p_t)^\gamma$ to the Cross-Entropy Loss. The Focal loss[3] is defined as:

$$\mathcal{L}_{\text{FL}}(p_t) = -(1 - p_t)^\gamma \log(p_t)$$

The gradient of the Focal loss[3] is:

$$\frac{d\mathcal{L}_{\text{FL}}(p_t)}{dx} = y(p_t - 1)^\gamma(\gamma p_t \log(p_t) + p_t - 1)$$

The parameter γ is a tunable focusing parameter that is adjusting the rate at which easy examples are downweighted. As γ increases, the model focuses more on hard, misclassified examples. When $\gamma = 0$, the Focal loss[3] becomes equivalent to the standard BCE Loss.

2.2.3 Class-Balanced Loss[4] :

Class Balanced Reweighing is a technique introduced by Cui et al. in their research paper. This method involves assigning different weights to the samples during training based on their class frequencies. The reweighing process is based on the inverse of each class' "effective" number of samples. The effective number of samples for a class is defined as a weighted average of the actual number of samples and a hyperparameter β. The parameter β controls the degree of reweighing and can be adjusted to balance majority and minority classes.

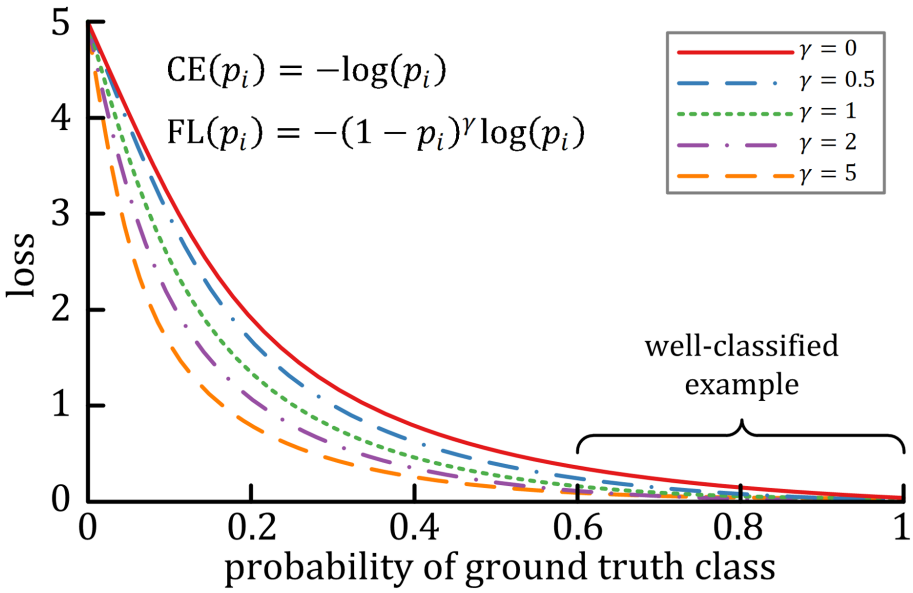


Figure 2.3: The cost of the well-classified cases and the modifying factor γ.

The authors assume that each data point occupies a 'volume' in the space where they are represented. When a large number of data points are present, they might overlap, and their total occupied volume is not necessarily the sum of the volumes of the individual points. Based on a simple argument, they derive the effective number of samples of a class that occupy as much volume as the existing set of points. The effective number of samples E_n is defined as:

$$E_n = \frac{1-\beta^{n_i}}{1-\beta}$$

where n_i is the number of samples from the i^{th} class.

They propose to introduce an additional balancing term in the cross-entropy loss function, which weighs the loss by the inverse of the effective number of samples of the class. The modified loss function is known as the CBRW loss[4].

$$Loss_{CBRW} = -\frac{1-\beta}{1-\beta^{n_i}} \cdot \log(p_i)$$

2.2.4 Balance Softmax[5] :

The purpose of Balance Softmax [5] is to remedy the disparity between the posterior distributions of training and testing data. The two conditional probabilities, ϕ for testing and $\hat{\phi}$ for training, are parameterized using the same model outputs, η.

Assuming that the usual Softmax function of the model output η is used to define ϕ, the expression for $\hat{\phi}$ amounts to:

$$\hat{\phi}_j = \frac{n_j e^{\eta_j}}{\sum_{k=1}^{N} n_k e^{\eta_k}}$$

The Balance Softmax[5] function is defined as:

$$\hat{l}(\theta) = -\log\left(\hat{\phi}_y\right) = -\log\left(\frac{n_y e^{\eta_y}}{\sum_{k=1}^{N} n_k e^{\eta_k}}\right)$$

2.2.5 LADE[6]:

The LADE (Label distribution DisEntangling) loss is a method used in deep learning to disentangle label representations, aiming to improve model performance by separating class information from other factors. It helps in learning more robust and discriminative features for classification tasks.

Label distribution DisEntangling (LADE)[6] loss is defined as follows:

$$\mathrm{L}_{\text{LADE-CE}}(f_\theta(x), y) = -\log(p_s(y|x;\theta))$$

$$= -\log\left(\frac{p_s(y) \cdot e^{f_\theta(x)[y]}}{\sum_c p_s(c) \cdot e^{f_\theta(x)[c]}}\right)$$

$$\mathrm{L}_{\text{LADE}}(f_\theta(x), y) = \mathrm{L}_{\text{LADE-CE}}(f_\theta(x), y) + \alpha \cdot \mathrm{L}_{\text{LADER}}(f_\theta(x), y)$$

where α is a nonnegative hyperparameter, which determines the regularization strength of $\mathrm{L}_{\text{LADER}}$.

2.3 Augmentation techniques

As we know, to enhance the performance of machine learning model, it is essential to perform data preprocessing steps before actually training the model. One crucial preprocessing step is Data Augmentation. In recent years, data augmentation has significantly improved model performance. Since these techniques enhance model performance, efforts have been made to further improve them.

Data augmentation methods complement loss-based methods. Unlike loss-based approaches, data augmentation techniques are applied to the input data while preparing it to be fed into the neural network. Essentially, these methods expand the training set by introducing additional novel examples derived from the existing data.

Data augmentation methods have evolved to enhance performance of machine learning models by generating new and varied training examples. Here are some notable techniques:

2.3.1 Mixup [7]:

The Mixup [7] technique, introduced by Zhang et al, creates new training examples by combining pairs of data points. Given two data points x_i and x_j with corresponding labels y_i and y_j, Mixup [7] generates a new data point $(\bar{x}, \bar{y})$ defined as:

$$\bar{x} = \lambda x_i + (1 - \lambda) x_j$$

$$\bar{y} = \lambda y_i + (1 - \lambda) y_j$$

where $\lambda \in (0, 1)$ and sampled from a Beta distribution Beta(α, α) with a parameter $\alpha > 0$. This approach encourages the model to generalize better by training on a convex combination of the input data points and their labels.

Figure 2.4: A comparison between two photos from different classes is shown by Mixup [7].

During training, the Mixup [7] process is applied as follows: for each data point in a training batch, another data point is randomly selected from the same batch and 'mixed' with the current point to produce a new data point. In effect, Mixup [7] encourages the idea that points lying on the line joining two data points should also possess a label that is a convex combination of the respective labels. This encourages the neural network to learn smooth decision boundaries between classes. By creating virtual examples with mixed labels, Mixup [7] reduces the inherent bias towards majority examples.

2.3.2 CutOut [8] :

CutOut [8] is a data augmentation technique where random patches of the input images are removed or "cut out" during the training process. This method forces the model to learn to recognize objects from the remaining parts of the image, thus making it more robust to occlusions and enhancing its ability to generalize from partial information.

By removing random patches, CutOut [8] helps in avoiding the model from becoming too reliant on particular features in the training data, encouraging it to learn

Figure 2.5: Original images: (left), CutOut (right) [8]

more distributed and generalized representations. This can lead to improved performance on tasks where objects may be partially occluded or presented in varying conditions.

The process of CutOut [8] is straightforward: for each image in the training set, a fixed-size patch is randomly selected and set to zero (or a constant value). This simple yet effective approach can notably improve the robustness and performance of model.

2.3.3 CutMix [9]

CutMix [9] performs data augmentation by randomly selecting a portion or patch of an input image and pasting it onto another image. The corresponding labels are also mixed based on the area ratio of the patches. This process encourages the model to learn from both the local and global features present in different images.

Figure 2.6: Original images: a(left) and b(middle), CutMix [9] of a and b (right)

It can be seen as a discrete version of Mixup [7] where certain portions of the image are completely replaced by another image while other parts remain the same. The area ratio of the patches determines the extent of mixing between the two images. The larger the patch, the stronger the mixing effect. This encourages the model to learn robust features that effectively recognise objects even when the

appearance of those objects is altered or occluded. CutMix [9] enables the model to learn locally discriminative features by combining patches from different images. This encourages the model to focus on local patterns and aids in the localisation of objects within an image.

2.3.4 CutThumbnail [10] :

CutThumbnail[10] is a data augmentation method that involves creating a smaller "thumbnail" version of an image and embedding it within the original image. This technique helps the model to focus on different scales and parts of the image, improving its ability to generalize from varying perspectives and partial views.

The process of CutThumbnail [10] involves the following steps:

1. A smaller thumbnail version of the original image is created by downscaling the image to a predefined size.

2. The thumbnail is then randomly placed within the original image, replacing the pixels at that location.

Figure 2.7: Original images: (left), CutThumbnail [10] (right)

This augmentation method encourages the model to learn to recognize objects at multiple scales and locations within the image, leading to better performance on tasks with varying object sizes and positions.

This augmentation method encourages the model to learn to recognize objects at multiple scales and locations within the image, leading to better performance on tasks with varying object sizes and positions.

By employing CutThumbnail [10], the model becomes more robust to variations in object scale and positioning, enhancing its overall performance and generalization capability.

2.3.5 Context-rich Minority Oversampling (CMO) [11]

One data augmentation technique designed to reduce class imbalance in datasets is context-rich Minority Oversampling (CMO) [11]. It makes better use of data from majority samples to contextualize minority samples. This entails combining foreground and background photographs from minority classes with images from majority classes.

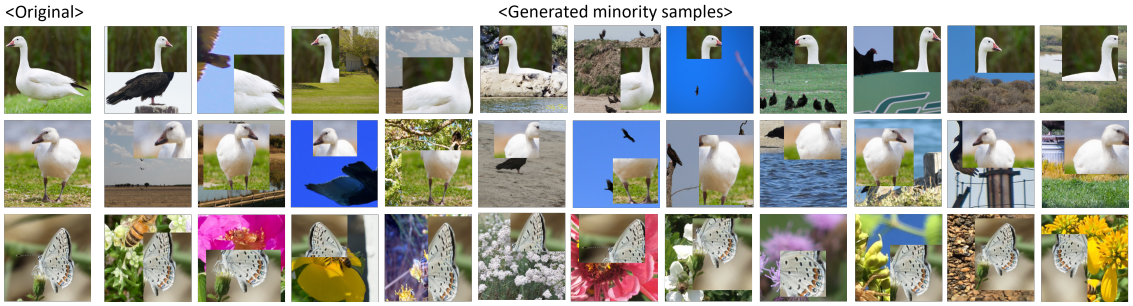


Figure 2.8: A presentation of CMO [11]'s minority graphics (minority categories: the Acmon blue (butterfly) and the snow geese). We choose created images at random for every starting image. Our methodology can generate rich contexts and different minority cases. In contrast, the produced images feature a variety of landscapes, such as the sky, sea, sand, and even a group of crows, while the original'snow goose' category only had pictures of a'snow goose' on grass. The model is able to obtain a robust representation of minority categories with the help of these recently created images.

The image combination is performed using the CutMix [9] technique, which mixes the images and their corresponding labels based on a predefined ratio. By sampling background images from the original data distribution and foreground images from a minority-class-weighted distribution, CMO [11] ensures that the augmented data promotes better generalization and performance for minority classes.

2.4 State-of-the-Art Two-Stage Approach for Imbalanced Data

Two-stage learning approaches are gaining traction for their effectiveness in handling imbalanced data. In the 1st stage, the model focuses on acquiring knowledge a robust feature representation of the data. The second stage is dedicated to training the classifier, leveraging the learned features to improve classification performance. This separation helps to address the challenges posed by class imbalance by ensuring that the feature learning process is not dominated by the majority classes.

Zhisheng Zhong, Jiequan Cui, Shu Liu, and Jiaya Jia recently made a major breakthrough in their article that was presented at the IEEE Conference on CVPR.

They proposed a two-stage framework called MiSLAS [20], which aims to improve recognition accuracy and mitigate over-confidence in models trained on long-tailed datasets. Their approach builds on existing techniques like LDAM-DRW [21] and decoupling models, incorporating strategies such as instance-balanced sampling and Mixup [7] to enhance representation learning and reduce over-confidence.

Despite these advancements, models trained on long-tailed datasets still suffer from **miscalibration** [15]. The networks typically to be overly confident in their predictions, especially for the majority classes, leading to a skewed probability distribution. This issue highlights the need for further research and refinement in calibration techniques to achieve more reliable and balanced model performance.

Chapter 3

Our Proposed Approaches

Long-tailed datasets provide networks that are more miscalibrated and overconfident [15] due to unequal ratio of component of each class. The SOTA(state-of-the-art) one-stage models and two-stage models also suffer from severe over-confidence. Despite these advancements, models trained on long-tailed datasets still experience miscalibration. The networks typically to be overly confident [15] in their expectations, especially for the majority classes, leading to a skewed probability distribution. This chapter discusses our attempt at a new technique to train a neural network in a class-imbalanced setting to improve miscalibration. We aim to design a novel algorithmic framework different from existing approaches but simple enough that it can be easily adopted and integrated into any existing ML pipeline.

3.1 Proposed Methods

In our investigation, we center our attention on remedying over-confidence challenges prevalent in long-tailed recognition, particularly in the realm of two-stage decoupling models. Our aim is to tackle these issues by employing soft label methodologies. Our strategy entails delving into the effectiveness of augmentation techniques and Adaptive Weight Adjustment to alleviate the complexities linked to over-confidence in long-tailed recognition settings.

- In Stage 1 of our study, we delve into the application of **Augmentation Techniques** to address over-confidence issues in long-tailed recognition. These techniques are aimed at enhancing the diversity and robustness of the training data, ultimately improving the model's ability to generalize across various classes.

- In Stage 2, we focus on **Adaptive Weight Adjustment** methods to further mitigate over-confidence in long-tailed recognition scenarios. These methods involve dynamically adjusting the weights assigned to different classes based

on their representation in the training data, thereby ensuring a more balanced and calibrated model.

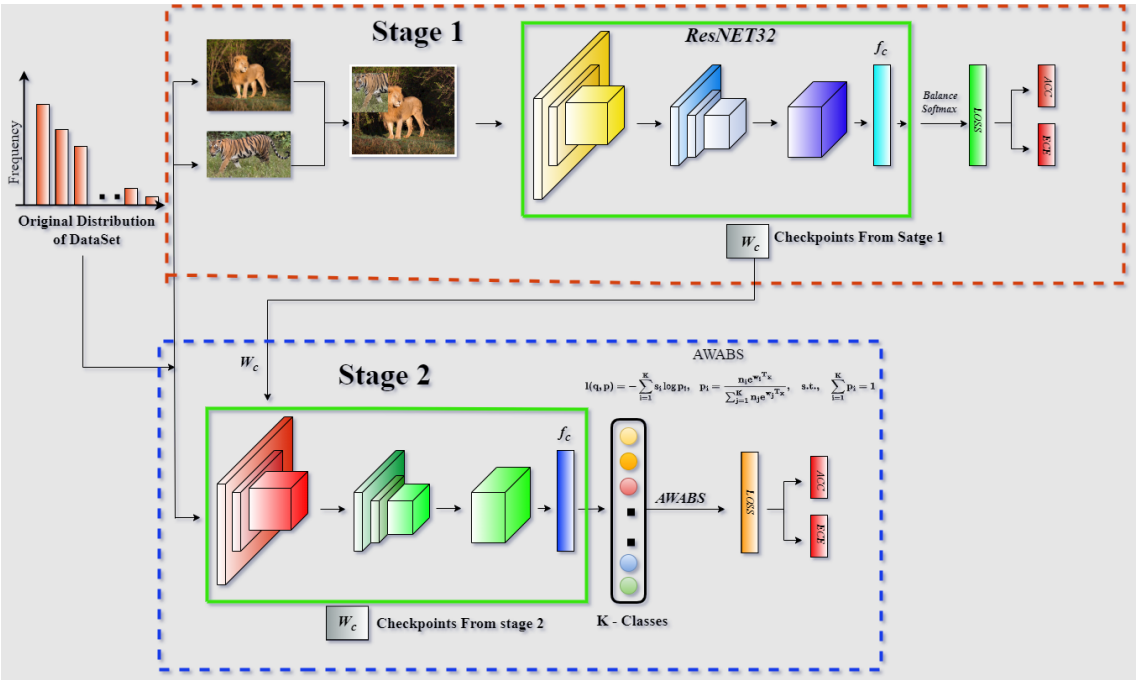


Figure 3.1: The framework of the proposed method.

3.2 Stage 1 : Augmentation techniques

In Stage 1 of our study, we focused on enhancing the robustness and generalization capability of the model through the application of data augmentation techniques. Specifically, we employed cutout, cutmix, and cutthumbnail augmentation methods during the re-training of the classifier. Following augmentation, the classifier was trained using a balanced softmax loss function to ensure fair treatment of samples from different classes.

Subsequently, we conducted an evaluation of the model's calibration performance, assessing the Expected Calibration Error (ECE) along with accuracy. This comprehensive analysis provided valuable insights into the model's calibration behavior across various segments of the class distribution, guiding further refinements in our calibration strategies to optimize performance across all classes.

3.3 Stage 2 : Adaptive Weight Adjustment

The Balanced Softmax[5] function for a particular class y is defined as:

$$p_y = \frac{n_y e^{\eta_y}}{\sum_{i=1}^{k} n_i e^{\eta_i}} = \frac{n_y e^{{w_y}^T x}}{\sum_{i=1}^{k} n_i e^{{w_i}^T x}}$$

- p_y represents the balanced probability for class y.
- n_y represents how many samples there are from class y.
- η_y represents the model output (logits) corresponding to class y.
- k represents number of classes.

Then the loss function is defined as :

$$l(y, p) = -log(p_y)$$

$$l(y, p) = -\log\left(\frac{n_y e^{{w_y}^T x}}{\sum_{i=1}^{k} n_i e^{{w_i}^T x}}\right) = -{w_y}^T x - \log(n_y) + \log(\sum_{i=1}^{k} n_i e^{{w_i}^T x})$$

In this context, we introduce AWABS (Adaptive Weight Adjustment with Balance Softmax) to address the problems of excessive faith in cross-entropy and the fluctuating distributions of anticipated likelihoods. It is formulated as:

$$l(q, p) = -\sum_{i=1}^{K} s_i \log p_i \tag{3.1}$$

where:

$$s_i = \begin{cases} 1 - \epsilon_y = 1 - f(N_y) & \text{if } i = y \\ \frac{\epsilon_y}{k-1} = \frac{f(N_y)}{k-1} & \text{if } i \neq y \end{cases} \tag{3.2}$$

$$p_y = \frac{n_y e^{{w_y}^T x}}{\sum_{i=1}^{k} n_i e^{{w_i}^T x}} \quad \text{and} \quad \sum_{i=1}^{k} p_i = 1 \tag{3.3}$$

where ϵ_y is a small label smoothing factor for Class-y, relating to its class number N_y.

The optimal solution is therefore:

$$w_i^{*T} x = \begin{cases} \log\left(\frac{K(1-\epsilon_y)}{\epsilon_y n_y}\right) + c, & \text{if } i = y, \\ c, & \text{otherwise.} \end{cases} \tag{3.4}$$

Here, c represents an arbitrary real number.

3.3.1 Proof : The solution of optimal solution 3.4

The loss function for K classes can be formulated as:

$$l = -\sum_{i=1}^{K} s_i \log p_i, \quad p_i = \text{softmax}(w_i^T x), \quad \text{s.t,} \quad \sum_{i=1}^{K} p_i = 1 \tag{3.5}$$

Here:

- p_i signifies the predicted probability assigned to class i.
- w_i denotes the weight parameter pertaining to the final fully-connected layer corresponding to class i.
- x stands for the input directed towards the final layer that is fully connected.

When the target label, s, is given as follows:

$$s_i = \begin{cases} 1; & \text{if } i = y \\ 0; & \text{if } i \neq y \end{cases}$$

where y represents the initial ground truth label. The generic loss function reduces to the widely used cross-entropy loss function.

When the target label s is defined as:

$$s_i = \begin{cases} 1 - \epsilon_y = 1 - f(N_y), & \text{if } i = y \\ \frac{\epsilon_y}{K-1} = \frac{f(N_y)}{K-1}, & \text{if } i \neq y \end{cases} \tag{3.6}$$

and

$$p_i = \frac{n_i e^{{w_i}^T x}}{\sum_{j=1}^{k} n_j e^{{w_j}^T x}} \tag{3.7}$$

then the **proposed** loss function becomes:

$$l(q,p) = -\sum_{i=1}^{K} s_i \log p_i, \quad p_i = \frac{n_i e^{{w_i}^T x}}{\sum_{j=1}^{k} n_j e^{{w_j}^T x}}, \quad \text{s.t.,} \quad \sum_{i=1}^{K} p_i = 1 \tag{3.8}$$

To get the optimal solution of equation 3.8 , we defined the **Lagrangian**. The equation is :

$$L = l + \lambda \left(\sum_{i=1}^{K} p_i - 1 \right) = -\sum_{i=1}^{K} s_i \log p_i + \lambda \left(\sum_{i=1}^{K} p_i - 1 \right) \tag{3.9}$$

where λ signifies the Lagrange multiplier.

Subsequently, by differentiating equation 3.9 concerning λ and p and contemplating the first-order conditions, we obtain:

$$\frac{\partial L}{\partial \lambda} = \sum_{i=1}^{K} p_i - 1 = 0$$

$$\frac{\partial L}{\partial p_i} = -\frac{s_i}{p_i} + \lambda = 0 \tag{3.10}$$

Solving equation 3.10 , we get

$$p_i = \frac{s_i}{\sum_{j=1}^{K} q_j}$$

The probability p_i is ascertained for the cross-entropy and re-weighting loss functions in the following manner:

$$p_i = \begin{cases} 1 & \text{if } i = y \\ 0 & \text{if } i \neq y \end{cases}$$

Now the given

$$p_i = \frac{n_i e^{{w_i}^T x}}{\sum_{j=1}^{k} n_j e^{{w_j}^T x}},$$

The optimal solution of $w_i^T x$ for both the re-weighting and cross-entropy loss functions yield same result: ${w_i^*}^T x = \infty$. This indicates that both loss functions tend to amplify the weight vector w_i corresponding to the correct class ($i = y$) significantly, while minimizing the weights of other classes ($j \neq y$). However, this approach fails to effectively modify the predicted probability distribution and alleviate overconfidence.

Now according to our proposed method solving equation 3.6 and equation 3.10 we get,

$$p_i = \frac{n_i e^{{w_i}^T x}}{\sum_{j=1}^{k} n_j e^{{w_j}^T x}} = \frac{s_i}{\sum_{j=1}^{K} q_j} = \begin{cases} 1 - \epsilon_y = 1 - f(N_y), & \text{if } i = y \\ \frac{\epsilon_y}{K-1} = \frac{f(N_y)}{K-1}, & \text{if } i \neq y \end{cases}$$

$$=> {w_i^*}^T x = \begin{cases} \log\left(\frac{K(1-\epsilon_y)}{\epsilon_y n_y}\right) + c, & \text{if } i = y, \\ c, & \text{otherwise.} \end{cases}$$

Proposed method provides a limited output, which is a specified real number c in the weight vector, as opposed to he infinitely best solution was discovered in the re-weighting and cross entropy approaches. This limited output helps to achieve a more complete result by precisely adjusting the expected distributions in the head, medium, and tail classes and successfully resolving overconfidence concerns.

In the long-tailed dataset, we arrange the labels in descending rank based on the quantity of instances, represented as $N_1 \geq N_2 \geq ... \geq N_K$. We propose a pe-

nalization strategy that strengthens label smoothing factors for classes with greater instance counts, since head classes contain a more varied set of examples and so have more promising projected probability than tail classes.

To achieve this, we introduce three types of related functions $f(N_y)$:

- Concave form

$$f(N_y) = \epsilon_K + (\epsilon_1 - \epsilon_K) \sin\left(\frac{\pi(N_y - N_K)}{2(N_1 - N_K)}\right) \tag{3.11}$$

-Linear form

$$f(N_y) = \epsilon_K + (\epsilon_1 - \epsilon_K)\frac{N_y - N_K}{N_1 - N_K} \tag{3.12}$$

-Convex form

$$f(N_y) = \epsilon_1 + (\epsilon_1 - \epsilon_K) \sin\left(\frac{3\pi}{2} + \frac{\pi(N_y - N_K)}{2(N_1 - N_K)}\right) \tag{3.13}$$

Here, ϵ_1 and ϵ_K are two hyperparameters. By setting $\epsilon_1 \geq \epsilon_K$, we ensure a decreasing trend in ϵ_1, leading to higher smoothing factors for classes with large instance numbers (N_y).Overconfidence is more common in head and middle classes than in tail classes, especially for large instance numbers. This change is intended to reduce overconfidence in these classes.

Chapter 4

Experimental Findings and Results

This chapter presents the results of our extensive experiments, along with our observations and interpretations. We have introduced a new method designed to enhance confidence calibration in long-tailed recognition. This method incorporates adaptive weight adjustment and augmentation techniques to better handle class imbalances. We have experimented with various datasets, fine-tuning hyperparameters, and assessing performance based on key metrics. Our methodology efficiently mitigates over-confidence and refines expected distributions across head, medium, and tail classes, as demonstrated by a comparison of our results with known methods in the literature.

4.1 Data and Configuration

For CIFAR10-LT and CIFAR100-LT, our experimental setup closely adheres to the recommendations given in [21], including implementation specifics and evaluation procedures. These datasets are well-suited for evaluating long-tailed recognition tasks, providing a robust benchmark for assessing the effectiveness of our proposed methods. We have adhered to standardized preprocessing steps, data augmentation techniques, and evaluation metrics to ensure comparability with existing work in the literature. The class imbalance ratios and other dataset-specific configurations have been maintained as per [21] to provide a consistent and fair evaluation environment.

4.1.1 Explanation of Datasets

The CIFAR10 and CIFAR100 datasets, which comprise 60k images each—50k for training and 10k for validation—are popular benchmarks. Ten categories make up CIFAR10, whilst 100 categories make up CIFAR100. Using the same parameters

as [21], we make use of the long tailed on these CIFAR datasets to guarantee a fair comparison. In order to do this, an imbalance factor $\beta = \frac{N_{\max}}{N_{\min}}$ is used to control the degree of data imbalance, where $N_{\min}$ and $N_{\max}$ stand for the amount of training samples for the least frequent classes and most respectively. We perform experiments employing IF(imbalance factors) of 200, 100, 50, and 10, adhering to the experimental protocols described by Cao et al. [21] and Zhou et al. [22].

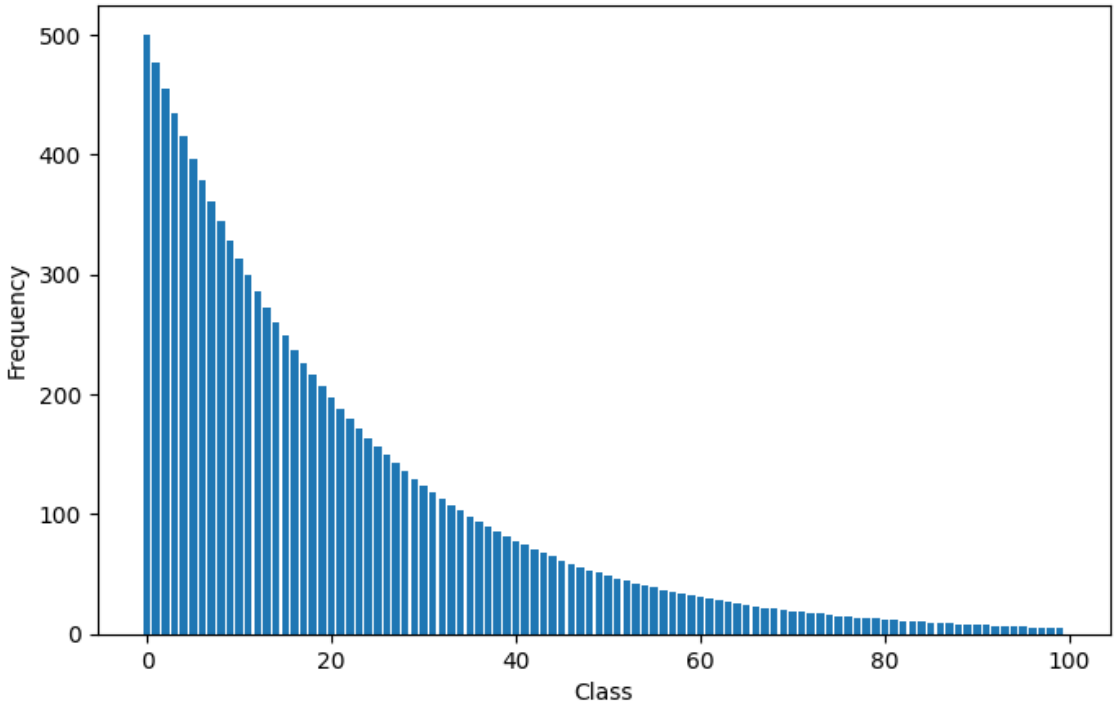


Figure 4.1: The Imbalanced CIFAR100 dataset's class distribution of data points

To further analyze the effects of imbalance, we categorize the classes into Head, Middle, and Tail classes based on their frequency. For CIFAR10, the head classes include classes indexed from 0 to 3, the middle classes include classes indexed from 3 to 7, and the tail classes include classes indexed from 7 to 10. Similarly, for CIFAR100, the head classes span indices from 0 to 30, the middle classes span indices from 30 to 70, and the tail classes span indices from 70 to 100. This categorization helps in understanding the performance of models across different levels of data frequency and highlights the challenges posed by imbalanced datasets.

4.1.2 Details of Implementation

The Stochastic Gradient Descent (SGD) optimizer with a momentum of 0.9 is employed in all experiments to optimize the networks. We mainly use the methodology of Cao et al.[21] for the CIFAR-LT datasets. Using the ResNet-32 [14] core on only one GPU, all proposed models are trained. We use a multistep rate of learning

schedule wherein at the 160th and 180th epochs of Stage-1, the learning rate is reduced by 0.1. This setup ensures that our training process is consistent and comparable with established methods in the literature, allowing for a robust evaluation of our proposed techniques.

4.2 Examination of Ablation

Calibration Performance:

Table 4.1 presents the calibration performance of our methods on CIFAR100-LT with an imbalance factor (IF) of 100. Our suggested methods greatly improve network calibration, delivering fewer Expected Calibration Errors (ECEs) than those using balanced datasets, as shown by the reliability diagrams with 15 bins. Furthermore, our techniques significantly enhance results for long-tailed recognition problems. Similar trends are observed on CIFAR10-LT, highlighting the effectiveness of our approach in improving calibration.

Table 4.1: outcomes of ResNet-32 trained with IF 100 on CIFAR100-LT

Dataset/Method	**Metrics**	
	Accuracy (ACC)	**Expected Calibration Error (ECE)**
Original CIFAR100	70.7	12.2
CIFAR100-LT, IF100	39.0	38.1
CIFAR100-LT,cRT [23]	40.7	29.5
CIFAR100-LT, LWS [24]	41.2	36.3
Mixup + cRT	45.2	13.8
Mixup + LWS	44.2	22.5
Mixup + LWS + Shifted BN	45.3	22.2
MiSLAS	47.0	4.83
Proposed Method 2	**48.03**	**1.96**

Training networks on unbalanced datasets leads to extreme over-confidence, according to all experimental outcomes. Softening the ground truth labels is a component of both conventional mixup and AWABS, indicating that training with rigid labels could potentially exacerbate network overconfidence. This observation underscores the necessity of our approach, which mitigates over-confidence by addressing label hardness and enhancing calibration in long-tailed scenarios.

Comparing Adaptive Weight Adjustment and re-weighting Now, we contrast the proposed adaptive weight adjustment, which employs balanced softmax [5], with re-weighting methods. The primary distinction lies in label transformation. In AWABS, the hard label undergoes a soft transition, adapting to the label distribution (as indicated in the non-target label scenario of Eq. 3.6: $s_i = \frac{f(N_y)}{K-1}, i \neq y$).

Conversely, re-weighting techniques omit this transformative step and directly set non-target labels to zero (i.e., $s_i = 0$).

Moreover, because of the label transformation, the optimal solution of $w_i^{*T}x$ in **AWABS** (Adaptive Weight Adjustment with Balance Softmax [5]) becomes Eq. 3.4. Conversely, the optimal solution of re-weighting mirrors that of cross-entropy ($w_i^{*T}x = \infty$), leading to ineffective adjustments in the predicted distribution and consequently, over-confidence. Our experimental results, as demonstrated in Table 4.5, reveal that incorporating the re-weighting method in Stage-2 leads to diminished performance and calibration in comparison to the **AWABS** approach.

Table 4.2: Comparing the proposed method's test accuracy (%)/ECE in Stage 2 with LAS, re-weighting, and CBCE [4]. These models were trained on CIFAR100-LT using IF 100, 50, and 10 and are all based on ResNet-32 [14].

Method / IF	**100**	**50**	**10**
LAS	47.0 / 4.83	52.3 / 2.25	63.2 / 1.73
CBCE [4]	44.3 / 20.2	50.5 / 19.1	62.5 / 13.9
Proposed method	**48.03 / 1.96**	**51.44 / 1.32**	**62.35 / 1.26**

4.3 Proposed Methods: Augmentation & AWABS

4.3.1 Proposed Method 1: CutOut & AWABS

In this proposed method, we outline a two-stage training process for improving model performance using CutOut and AWABS techniques. During Stage 1, the model is trained using the CutOut data augmentation method combined with BalanceSoftmax loss to enhance the robustness and balance of the model. For each epoch, the loss is computed and back-propagated over batches, followed by the computation of accuracy (ACC) and Expected Calibration Error (ECE) on the validation set, with periodic checkpointing.

Algorithm 1 Proposed Method

```
// Stage 1: Training with CutOut and BalanceSoftmax
for j ← 1 to stage1_epochs do
  loss ← 0.0
  for (images, labels) in current batch do
    outputs ← CutOut(images)
    batch_loss ← BalanceSoftmax(outputs, labels)
    loss ← loss + batch_loss
    Back-propagate the loss
  end for
  ACC ← ComputeAccuracy(model, validation_data)
  ECE ← ComputeECE(model, validation_data)
  SaveCheckpoint(model, optimizer, epoch)
end for
// Stage 2: Training with AWABS
LoadCheckpoint(model, optimizer, stage1_checkpoint)
for j ← 1 to stage2_epochs do
  loss ← 0.0
  for (images, labels) in current batch do
    outputs ← model(images) // Use the model loaded from stage 1
    batch_loss ← AWABS(outputs, labels)
    loss ← loss + batch_loss
    Back-propagate the loss
  end for
  ACC ← ComputeAccuracy(model, validation_data)
  ECE ← ComputeECE(model, validation_data)
  SaveCheckpoint(model, optimizer, epoch)
end for
return ACC, ECE
```

In Stage 2, the model continues to train with AWABS, an adaptive weight adjustment technique, ensuring further fine-tuning and stability. Similarly, loss computation, accuracy, and ECE measurements, along with checkpointing, are performed iteratively. The process concludes by returning the final accuracy and ECE metrics.

Table 4.3: Results from Proposed Method 1

Dataset	**Stage 1 (ACC / ECE)**	**Stage 2 (ACC / ECE)**	**Head**	**Middle**	**Tail**
CIFAR10-LT, IF=10	89.870 / 8.628	90.010 / 1.534	93.9	86.575	90.167
CIFAR10-LT, IF=50	85.410 / 8.137	84.770 / 2.162	92.167	82.55	80.333
CIFAR10-LT, IF=100	83.330 / 7.232	81.950 / 3.696	92.167	80.75	73.333
CIFAR100-LT, IF=10	61.280 / 8.973	62.400 / 1.223	66.778	63.914	55.138
CIFAR100-LT, IF=50	52.140 / 5.986	52.430 / 2.120	63.806	53.629	36.862
CIFAR100-LT, IF=100	47.290 / 3.851	48.130 / 6.376	64.056	49.686	26.483
CIFAR100-LT, IF=200	43.410 / 4.071	45.910 / 2.668	56.722	47.286	30.828

4.3.2 Proposed Method 2: CutMix & AWABS

In this method , we evaluate the efficiency of CutOut [8] in enhancing both the localizability and generalizability of trained models across multiple tasks. We begin by examining its impact on image classification and weakly supervised object localization, followed by an exploration of its transferability to tasks like object detection and image captioning. Furthermore, we demonstrate how CutMix [9] enhances model robustness and alleviates issues related to over-confidence.

Algorithm 2 : Proposed Method 2

```
// Stage 1: Training with CutMix and BalanceSoftmax
for j ← 1 to stage1_epochs do
    loss ← 0.0
    for (images, labels) in current batch do
        outputs ← CutMix(images)
        batch_loss ← BalanceSoftmax(outputs, labels)
        loss ← loss + batch_loss
        Back-propagate the loss
    end for
    ACC ← ComputeAccuracy(model, validation_data)
    ECE ← ComputeECE(model, validation_data)
    SaveCheckpoint(model, optimizer, epoch)
end for
// Stage 2: Training with AWABS
LoadCheckpoint(model, optimizer, stage1_checkpoint)
for j ← 1 to stage2_epochs do
    loss ← 0.0
    for (images, labels) in current batch do
        batch_loss ← AWABS(outputs, labels)
        loss ← loss + batch_loss
        Back-propagate the loss
    end for
    ACC ← ComputeAccuracy(model, validation_data)
    ECE ← ComputeECE(model, validation_data)
    SaveCheckpoint(model, optimizer, epoch)
end for
return ACC, ECE
```

In Stage 2, we utilize Stage 1 checkpoints, apply AWABS, compute loss with adjusted weights, and re-evaluate accuracy and ECE. Additionally, we provide performance metrics for different class groups (Head, Middle, Tail), illustrating how AWABS promotes a more balanced accuracy distribution across various difficulty levels of classes.

The table below presents the results of the Proposed Method 2: CutOut [8] & AWABS.

Table 4.4: Results from Proposed Method 2

Dataset	**Stage 1 (ACC / ECE)**	**Stage 2 (ACC / ECE)**	**Head**	**Middle**	**Tail**
CIFAR10-LT, IF=10	90.710 / 7.553	90.590 / 1.631	93.333	87.725	91.667
CIFAR10-LT, IF=50	86.530 / 5.840	85.570 / 2.556	92.433	83.275	81.767
CIFAR10-LT, IF=100	82.810 / 5.228	81.220 / 2.553	92.133	79.625	72.433
CIFAR100-LT, IF=10	62.170 / 4.157	62.350 / 1.259	66.056	61.971	58.207
CIFAR100-LT, IF=50	51.220/ 1.395	51.440 / 1.321	60.833	53.057	36.345
CIFAR100-LT, IF=100	47.850 / 2.414	48.030 / 1.961	61.556	48.429	27.724
CIFAR100-LT, IF=200	43.420 / 3.194	44.930 / 1.286	56.111	44.829	31.034

4.3.3 Proposed Method 3 : Cut-Thumbnail & AWABS

In this method, we evaluate the performance of Cut-Thumbnail [10] in enhancing the localizability and generalizability of trained models across various tasks. We start by analyzing its impact on classifying images and localizing objects with poor supervision, followed by its transferability to tasks such as object detection and image captioning.

Additionally, we demonstrate how Cut-Thumbnail [10] improves model robustness and reduces over-confidence. In Stage 2, we use the checkpoints from Stage 1, apply Adaptive Weight Adjustment with Balance Softmax (AWABS), compute the loss with adjusted weights, and re-evaluate accuracy and Expected Calibration Error (ECE). Furthermore, we provide performance metrics for different class groups (Head, Middle, Tail), showing how AWABS achieves a more balanced accuracy distribution across classes of varying difficulty levels.

Algorithm 3 : Proposed Method 3

```
// Stage 1: Training with Cut-Thumbnail and BalanceSoftmax
for j ← 1 to stage1_epochs do
  loss ← 0.0
  for (images, labels) in current batch do
    outputs ← Cut-Thumbnail(images)
    batch_loss ← BalanceSoftmax(outputs, labels)
    loss ← loss + batch_loss
    Back-propagate the loss
  end for
  ACC ← ComputeAccuracy(model, validation_data)
  ECE ← ComputeECE(model, validation_data)
  SaveCheckpoint(model, optimizer, epoch)
end for
// Stage 2: Training with AWABS
LoadCheckpoint(model, optimizer, stage1_checkpoint)
for j ← 1 to stage2_epochs do
  loss ← 0.0
  for (images, labels) in current batch do
    batch_loss ← AWABS(outputs, labels)
    loss ← loss + batch_loss
    Back-propagate the loss
  end for
  ACC ← ComputeAccuracy(model, validation_data)
  ECE ← ComputeECE(model, validation_data)
  SaveCheckpoint(model, optimizer, epoch)
end for
return ACC, ECE
```

The table below presents the results of the Proposed Method 2: Cut-Thumbnail [10] & AWABS.

4.4 Effect of ϵ_1 and ϵ_K on AWABS

In a recognition system, the classifier normally classifies the input as Class-y if class-y has a higher than 0.5 expected probability.. To ensure this behavior remains reasonable under label-aware smoothing, we maintain the constraint $0 \leq \epsilon_K \leq \epsilon_1 \leq 0.5$ in Eqs 3.11 , 3.12 and 3.13. This constraint ensures that the smoothing does not overly diminish the classifier's confidence in its top predictions while still providing a measure of uncertainty that can improve generalization and robustness to noise.

By controlling these hyperparameters, we can effectively manage the trade-off between model confidence and uncertainty, leading to more robust and reliable performance across a variety of tasks and datasets.

Table 4.5: Results from Proposed Method 3

Dataset	**Stage 1 (ACC / ECE)**	**Stage 2 (ACC / ECE)**	**Head**	**Middle**	**Tail**
CIFAR10-LT, IF=10	90.190 / 13.294	90.500 / 2.371	93	87.55	91.667
CIFAR10-LT, IF=50	85.940 / 10.954	85.840 / 2.717	91.4	83.725	83.1
CIFAR10-LT, IF=100	83.360 / 13.774	82.460 / 3.527	90.3	80.525	77.2
CIFAR100-LT, IF=10	61.440 / 8.037	61.320 / 2.113	65.583	61.371	55.966
CIFAR100-LT, IF=50	50.790 / 6.812	50.500 / 2.895	60.528	51.257	37.138
CIFAR100-LT, IF=100	46.620 / 5.527	46.690 / 4.631	59.528	47	27.448
CIFAR100-LT, IF=200	43.130 / 2.941	44.470 / 2.460	58.028	43.086	28.897

4.5 Effect of AWABS on Prediction Distribution

To understand the impact of Adaptive Weight Adjustment with Balance Softmax (AWABS) on prediction distributions, we train two Label-Aware Smoothing (LWS) models on the CIFAR100(LT) dataset with an IF of 100: one using cross-entropy loss and the other using AWABS.

Reduced Over-Confidence: AWABS reduces over-confidence in head and medium classes, balancing prediction distributions. Improved Tail Class Predictions: Tail class predictions shift right, indicating better confidence and accuracy for these classes.

4.6 Comparison with SOTA (State-of-the-Arts)

To assess the efficiency of our proposed method, we conducted a thorough comparison with both one stage and two stage methods in domain of long tail recognition.

In our evaluation, we scrutinized the performance against the following one stage models: CE, LDAM Loss [21], and LADE [6]. Additionally, we benchmarked our approach against various two stage methods, including LWS [24], cRT [23] and MisLAS [20] . To ensure equitable comparisons, we integrated augmentation techniques and AWABS into our method. Moreover, we directly juxtaposed our method with BBN [22], which utilizes double samplers and is trained akin to a mixup approach. This comparison provided valuable insights into the relative efficacy of our proposed approach compared to BBN [22]

Table 4.6: ResNet 32 based models, trained on CIFAR10-LT, top-1 accuracy (%) / ECE.

Method	**CIFAR10-LT**		
IF	100	50	10
CE	70.4	74.8	86.4
mixup[7]	73.1	77.8	87.1
BBN [22]	79.9	82.2	88.4
LDAM+DRW[21]	77.1	81.1	88.4
DRW+REMIX [25]	79.8	-	89.1
cRT+mixup [23]	79.1 / 10.6	84.2 / 6.89	-
LWS+mixup	76.3 / 15.6	82.6 / 11.0	89.6 / 5.41
MiSLAS[20]	82.1 / 3.70	85.7 / 2.17	90.0 / 1.20
Proposed Method 1	81.950 / 3.696	84.770 / 2.162	90.010 / 1.534
Proposed Method 2	**82.2/2.55**	85.57/2.56	**90.59/1.63**
Proposed Method 3	**82.460 / 3.527**	**85.840 / 2.717**	**90.500 / 2.371**

Table 4.7: ResNet-32 based models , trained on CIFAR100LT, top-1 accuracy (%) / ECE

Method	**CIFAR100-LT**			
IF	200	100	50	10
CE	36.5	38.4	43.9	55.8
mixup[7]	-	39.6	45.0	58.2
LDAM+DRW[21]	38.5	42.1	46.7	58.8
BBN[22]	-	42.6	47.1	59.2
LADE[6]	-	45.4 / 2.54	50.5 / 0.148	61.7 /1.46
DRW+REMIX [25]	38.45	46.8	46.62	61.3
cRT+mixup [23]	40.16	45.1 / 13.8	50.9 / 10.8	62.1 / 6.83
LWS+mixup	44.9	44.2 / 22.5	50.7 / 19.2	62.3 / 13.4
MiSLAS[20]	43.53	47.0 / 4.83	52.3 / 2.25	63.2 / 1.73
Proposed Method 1	45.910 / 2.668	48.130 / 6.376	52.430 / 2.120	62.400 / 1.223
Proposed Method 2	**45.930 / 1.286**	**48.03/1.96**	**51.44/1.32**	**62.35/1.26**
Proposed Method 3	44.470 / 2.460	46.690 / 4.631	50.500 / 2.895	61.320 / 2.113

4.7 Findings from CIFAR Long Tailed experiments

We conducted comprehensive testing on CIFAR10(LT) and CIFAR100(LT) datasets with imbalance factors (IF) of 100, 50, and 10, following the same settings as previous studies [20]. The results, summarized in Tables 4.6 and 4.7, exhibit that **Proposed Method 2** significantly outshines most previous methods in both top-1 accuracy and Expected Calibration Error (ECE).

4.7.1 Comprehensive Report on CIFAR100-LT Classification Performance

In our evaluation of various classification methods on the CIFAR100-LT dataset, we observed significant differences in performance across different imbalance factors (IF). The baseline method, **Cross-Entropy (CE)**, showed lower accuracy across all IFs. Enhancements such as **mixup** [7] and **LDAM-DRW** [21] improved performance, with notable gains in accuracy. Advanced techniques like **BBN** [22] and **Remix-DRW** [25] further boosted accuracy, particularly at higher imbalance factors. Calibration methods, including **cRT+mixup** [23]and **LWS** [24], provided a balance between accuracy and calibration error (**ECE**), demonstrating better-calibrated models. The **MiSLAS** [20] method excelled in both accuracy and calibration, particularly at higher IFs.

Our proposed methods demonstrated superior performance:

- **Proposed Method 1** achieved robust accuracy and low **ECE** values across all IFs.
- **Proposed Method 2** stood out with the best overall performance, combining high accuracy with low **ECE**, particularly excelling at IF 100 and IF 10.
- **Proposed Method 3** also performed exceptionally well, demonstrating high accuracy and effective calibration across all IFs.

4.7.2 Comprehensive Report on CIFAR100-LT Classification Performance

In our evaluation of different classification methods on the CIFAR100-LT dataset, we observed significant differences in performance across different imbalance factors (IF). The baseline method, **Cross-Entropy (CE)**, showed lower accuracy across all IFs. Enhancements such as **mixup** [7] and **LDAM+DRW** [21] improved performance, with notable gains in accuracy. Advanced techniques like **BBN** [22] and **Remix+DRW** [25] further boosted accuracy, particularly at higher imbalance factors. Calibration methods, including **cRT+mixup** [23] and **LWS+mixup**, provided a balance between accuracy and calibration error (ECE), demonstrating better-calibrated models. The **MiSLAS** [20] method excelled in both accuracy and calibration, particularly at higher IFs.

Our proposed methods demonstrated superior performance:

- **Proposed Method 1** achieved robust accuracy and low ECE values across all IFs.

- **Proposed Method 2** stood out with the best overall performance, combining high accuracy with low ECE, particularly excelling at IF 200, 100 and IF 10.
- **Proposed Method 3** also performed exceptionally well, demonstrating high accuracy and effective calibration across all IFs.

4.7.3 Performance on different splits of classes:

Table 4.8: Performance comparison of different methods on CIFAR100-LT with IF 100, evaluating accuracy across three class splits: Head, Middle, and Tail.

Method	**Head**	**Middle**	**Tail**	Top 1% ACC
MiSLAS [20]	63.3	46.7	22.7	47.0
LWS	61.4	43.4	19.6	44.2
cRT [23]	65.91	43.74	23.8	45.1
Proposed Method 1	64.056	49.686	26.483	48.13
Proposed Method 2	**61.556**	**48.429**	**27.724**	**48.03**
Proposed Method 3	59.528	47	27.448	46.69

Table 4.8 compares the accuracy of various methods on CIFAR100-LT with an imbalance factor of 100 across Head, Middle, and Tail classes.

MiSLAS [20]: Best on Head (63.3), good overall (47.0). **LWS** [24]: Lower performance across all splits, overall (44.2). **cRT** [23]: Strong Head (65.91), moderate overall (45.1). **Proposed Method 1**: Balanced, highest overall (48.13). **Proposed Method 2**: Best Tail (27.724), strong overall (48.03). **Proposed Method 3**: Competitive, especially in Tail (46.69).

Proposed methods excel, particularly in Tail and Middle classes, showing better handling of imbalanced data.

4.8 Summary of Proposed Methods

In this performance comparison on CIFAR100-LT with an IF of 100, across three class splits (Head, Middle, and Tail), our **Proposed Method 2** emerges as a standout performer with accuracy values of **45.930%**, **48.030%**, and **51.440%** for the Head class, Middle class, and Tail classes, respectively. Additionally, the Expected Calibration Error (ECE) values for Proposed Method 2 are **1.286**, **1.960**, **1.320**, and **1.260** for the corresponding splits.

This suggests the effectiveness of **Proposed Method 2** in addressing class imbalance across different categories within the dataset while maintaining reliable calibration. While other methods also demonstrate competitive performance in terms of accuracy, the consistent calibration performance of **Proposed Method 2**

underscores its potential for improving model reliability in long-tailed datasets like CIFAR100-LT.

In summary, the strong performance of **Proposed Method 2** in both accuracy and calibration metrics highlights its effectiveness and suitability for real-world deployment. These results emphasize the significance of our proposed method in addressing class imbalance and improving model reliability in long-tailed distributions.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

To conclude the comparison between various ResNet-32 based models trained on CIFAR10-LT and CIFAR100-LT datasets, we observe significant variations in performance metrics, particularly in terms of top-1 accuracy and ECE. Let's highlight some key findings:

CIFAR10-LT Dataset:

- The proposed Method 2 consistently outperforms other methods across all imbalance factors (IF) with the highest accuracy and lowest ECE. This indicates its robustness in handling class imbalances.
- Method 3 also demonstrates competitive performance, especially in maintaining low ECE values across different IFs.
- MiSLAS [20] achieves the highest accuracy on CIFAR10-LT, closely followed by the proposed Method 2.
- Both Proposed Method 1 and Method 2 exhibit strong accuracy and calibration metrics, indicating their effectiveness in addressing long-tail class distributions.

CIFAR100-LT Dataset:

- Once again, the **Proposed Method 2** stands out as the top performer, showcasing superior accuracy and calibration metrics across various imbalance factors.
- LADE and MiSLAS [20] methods also perform well but are surpassed by the **Proposed Method 2** in terms of both accuracy and ECE.

- Notably, **Proposed Method 2** achieves the lowest ECE values across all imbalance factors, indicating its excellent calibration.

Overall Highlights:

- **Consistency**: The **Proposed Method 2** consistently outperforms or matches other methods in terms of accuracy and calibration on both CIFAR10-LT and CIFAR100-LT datasets.
- **Robustness**: **Proposed Method 2** shows robustness across different levels of class imbalance, indicating its applicability in various real-world scenarios.
- **Calibration**: The proposed methods exhibit superior calibration, crucial for reliable uncertainty estimation in classification tasks.
- **Competitive Performance**: While MiSLAS [20] and LADE [6] demonstrate competitive performance, the proposed Method 2 consistently achieves the best balance between accuracy and calibration.
- Analyzing model performance across different class splits, such as Head, Middle, and Tail categories, reveals insights into their ability to handle class imbalances. MiSLAS [20] excels on frequent classes but struggles with rare ones, while Proposed Method 2 stands out by demonstrating high accuracy on rare categories, indicating its robustness in handling imbalanced data comprehensively. This comparative analysis underscores the significance of evaluating models across diverse class distributions to gauge their effectiveness in real-world scenarios.

In summary, the results suggest that the proposed Method 2 presents a promising approach for addressing class imbalances in classification tasks, offering superior performance in terms of accuracy and calibration across both CIFAR10-LT and CIFAR100-LT datasets.

5.2 Future Work

Looking ahead, there are several exciting avenues for future research that can further enhance the capabilities of our proposed methods:

5.2.1 Exploration of Mixed Cut Thumbnails

Integrating image cropping techniques with CutOut augmentation presents an intriguing opportunity to improve model robustness and generalization. Investigating

the synergies between these techniques could lead to significant performance gains across various datasets and scenarios.

5.2.2 Dynamic Balance Softmax Loss Functions

Dynamic balance softmax loss functions hold promise in better handling class imbalances by adapting to the data distribution dynamically. Exploring novel loss formulations tailored to specific datasets and imbalance factors could unlock new possibilities for improving model performance.

5.2.3 Implementation of a Reverse Sampler

The implementation of a reverse sampler, prioritizing samples from underrepresented classes during training, could alleviate class imbalance challenges. By ensuring sufficient exposure to minority classes, this technique has the potential to enhance model generalization and performance on long-tailed datasets.

Adapting to Other Long-Tailed Datasets: Extending our proposed methods to other long-tailed datasets beyond CIFAR100-LT can validate their generalizability and effectiveness in diverse real-world scenarios. Understanding how these techniques perform across different domains and data distributions is crucial for advancing the field of class imbalance mitigation.

Bibliography

[1] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," 2017.

[2] Y. Ho and S. Wookey, "The real-world-weight cross-entropy loss function: Modeling the costs of mislabeling," *IEEE Access*, vol. 8, p. 4806–4813, 2020.

[3] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," 2018.

[4] Y. Cui, M. Jia, T.-Y. Lin, Y. Song, and S. Belongie, "Class-balanced loss based on effective number of samples," in *CVPR*, 2019.

[5] J. Ren, C. Yu, S. Sheng, X. Ma, H. Zhao, S. Yi, and H. Li, "Balanced meta-softmax for long-tailed visual recognition," in *Proceedings of Neural Information Processing Systems(NeurIPS)*, Dec 2020.

[6] Y. Hong, S. Han, K. Choi, S. Seo, B. Kim, and B. Chang, "Disentangling label distribution for long-tailed visual recognition," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 6626–6636, 2021.

[7] Y. N. D. D. L.-P. Hongyi Zhang, Moustapha Cisse, "mixup: Beyond empirical risk minimization," *International Conference on Learning Representations*, 2018.

[8] T. DeVries and G. W. Taylor, "Improved regularization of convolutional neural networks with cutout," *arXiv preprint arXiv:1708.04552*, 2017.

[9] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo, "Cutmix: Regularization strategy to train strong classifiers with localizable features," in *International Conference on Computer Vision (ICCV)*, 2019.

[10] T. Xie, X. Cheng, X. Wang, M. Liu, J. Deng, T. Zhou, and M. Liu, "Cut-thumbnail: A novel data augmentation for convolutional neural network," in *Proceedings of the 29th ACM International Conference on Multimedia*, MM '21, (New York, NY, USA), p. 1627–1635, Association for Computing Machinery, 2021.

[11] S. Park, Y. Hong, B. Heo, S. Yun, and J. Y. Choi, "The majority can help the minority: Context-rich minority oversampling for long-tailed classification," 2022.

[12] G. V. Horn, O. M. Aodha, Y. Song, Y. Cui, C. Sun, A. Shepard, H. Adam, P. Perona, and S. Belongie, "The inaturalist species classification and detection dataset," 2018.

[13] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba, "Places: A 10 million image database for scene recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 6, pp. 1452–1464, 2018.

[14] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

[15] S. Ao, S. Rueger, and A. Siddharthan, "Two sides of miscalibration: Identifying over and under-confidence prediction for network calibration," 2023.

[16] T. Hasanin and T. Khoshgoftaar, "The effects of random undersampling with simulated class imbalance for big data," in *2018 IEEE International Conference on Information Reuse and Integration (IRI)*, pp. 70–79, 2018.

[17] A. Ghazikhani, H. S. Yazdi, and R. Monsefi, "Class imbalance handling using wrapper-based random oversampling," in *20th Iranian Conference on Electrical Engineering (ICEE2012)*, pp. 611–616, 2012.

[18] N. A. Azhar, M. S. M. Pozi, A. M. Din, and A. Jatowt, "An investigation of smote based methods for imbalanced datasets with data complexity analysis," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 7, pp. 6651–6672, 2023.

[19] Z. Xu, D. Shen, T. Nie, Y. Kou, N. Yin, and X. Han, "A cluster-based oversampling algorithm combining smote and k-means for imbalanced medical data," *Information Sciences*, vol. 572, pp. 574–589, 2021.

[20] S. L. Zhisheng Zhong, Jiequan Cui and J. Jia, "Improving calibration for long-tailed recognition," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[21] K. Cao, C. Wei, A. Gaidon, N. Arechiga, and T. Ma, "Learning imbalanced datasets with label-distribution-aware margin loss," in *Advances in Neural Information Processing Systems*, 2019.

[22] B. Zhou, Q. Cui, X.-S. Wei, and Z.-M. Chen, "BBN: Bilateral-branch network with cumulative learning for long-tailed visual recognition," pp. 1–8, 2020.

[23] B. Kang, S. Xie, M. Rohrbach, Z. Yan, A. Gordo, J. Feng, and Y. Kalantidis, "Decoupling representation and classifier for long-tailed recognition," 2020.

[24] Y. Cui, M. Jia, T.-Y. Lin, Y. Song, and S. Belongie, "Class-balanced loss based on effective number of samples," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9260–9269, 2019.

[25] H.-P. Chou, S.-C. Chang, J.-Y. Pan, W. Wei, and D.-C. Juan, "Remix: Rebalanced mixup," 2020.

[26] S. Park and P. M. Pardalos, "Deep data density estimation through donsker-varadhan representation," 2021.