Exploring Deep Learning Techniques for Long-Tailed Recognition: Methods, Models, and Analysis

Master's Thesis in
Electrical Engineering

By

Christine Annelise Midtgaard
au521655
Study Program: Electrical Engineering

Supervisor
Kim Bjerge
Associate Professor
kbe@ece.au.dk

M.Sc. in Electrical Engineering
Department of Engineering Science & Technology
Aarhus University

January 3, 2025

Abstract

This thesis, titled *Exploring Deep Learning Techniques for Long-Tailed Recognition: Methods, Models, and Analysis*, by Christine Annelise Midtgaard, explores the challenges and solutions related to training deep learning models on long-tailed datasets. Long-tailed datasets, where a few classes dominate with abundant samples while many classes have sparse representation, pose significant challenges for traditional training methods. These imbalances often lead to models that perform well on majority classes but struggle to recognize or generalize to minority (tail) classes.

This thesis focuses on evaluating and implementing state-of-the-art methods for long-tailed learning, as outlined in the survey Deep Long-Tailed Learning: A Survey by Zhang et al. The methodologies explored include advanced sampling strategies, re-weighted loss functions, and modifications to deep learning architectures tailored to imbalanced data.

A unique application of these methods is demonstrated on a custom dataset of moth images collected near the equator, where the goal is accurate species identification. Through a series of experiments, the thesis investigates how different approaches to long-tailed learning impact model performance across head, middle, and tail classes.

The findings contribute to understanding the efficacy of these methods and provide insights into best practices for handling real-world long-tailed datasets.

Acknowledgements

I would like to thank my supervisor, Kim Bjerge, for their guidance, constructive feedback, and inspiring conversations throughout the development of this thesis. I am also deeply thankful to my family for providing me with room for my frustrations, and for cooking meals during the challenging times of this journey. I want to thank my friend, Line, for her valuable insights in writing a thesis. A special thanks to boyfriend for his outstanding patience and understanding.

Contents

Chapter 1

Introduction

Image classification is one of the main challenges computer vision.
Deep learning has become a prominent solution in recent years for tackling image regocnision tasks. With the availability of large datasets, i.e. ImageNet, along with GPUs, training of deep learning models have become easier, and have led to remarkable results [reference here]. The trained models have shown image classification with accuracies of over 80 % [reference here], and hence the interest in deep learning for image recognition is high. However, the high accuracies are for on models trained on balanced datasets with thounsands of samples per class [reference here], while most real-world datasets are skewed in samples per class [reference here].
This thesis focuses on the problem with long-tailed datasets. The problem with training a deep learning model on long-tailed datasets is that the model will effectively the data from the classes with most samples, and not the classes with few samples. The finsihed model will then not recognize an input from the tail classes. Most real-world datasets follows a long-tailed structure, hence the need for a reliable method to detect examples of tail-class data. The aim of this thesis is to try out some of the methods tackling the long-tailed problem for deep learning described in the paper *Deep Long-Tailed Learning: A Survey* by Zhang et al.[1] to find a method for long-tailed learning that works on a specific long-tailed dataset of images of moths taken around equator. The goal of the moth dataset is to identify species.

1.1 Problem Definition

Something about head and tail classes.
The goal of this project is to investigate deep learning models, like Convolutional Neural Networks (CNNs) and Vision Transformers (ViTs), and methods, mainly class re-balancing through cost-sensitive learning, and analyze their performance on datasets representaing a long-tailed structure.
Four models are investigated, where three of them are CNNs, and one is a ViT. The CNNs investigated are the MobileNetV2, ResNet50v2, and ConvNeXt Base, while the ViT investigated is the ViT-B/16. All four models are pretrained on Im-

ageNet, and further trained on both a balanced version of CIFAR100 and an long-tailed version of CIFAR100, called CIFAR100-LT, and introducing the long-tailed technique as class re-balancing in the loss functions by using re-weighting. All models are trained with Softmax Cross-Entropy Loss, Weighted Softmax Cross-Entropy Loss, Focal Loss, Class-Balanced Loss, Balanced Softmax Loss, LDAM Loss, and Equalization Loss. The main purpose of this master's thesis is to compare class re-balancing methods on a well respresenting type of models used in deep learning image recognition tasks. Furhter comparison with state-of-the-art methods for deep long-tailed learning will be made. The motivation for this comparison is to find a way to train on a real-world long-tailed dataset of moths to correctly identify species.

1.1.1 Goals of this thesis

The goals of this thesis are to:

1. Investigate the efficacy of long-tailed learning methods by assessing their performance on tail classes without sacrificing accuracy on head classes.

2. Understand how model design affects the performance of deep long-tailed learning methods.

3. Provide a comprehensive insight in comparisons that inform the choice of methods for long-tailed distributions in acedemic and industry settings.

1.1.2 Hypothesis

Deep long-tailed learning methods, such as the carefully designed loss functions tailored for long-tailed ditributed datasets, can improve the performance of underrepresented (tail) classes while maintaining the overall accuracy across diverse model architectures. The effectiveness of these loss functions is influenced by the choise of model architecture and the degree of imbalance of the dataset.

1.1.3 Approach

This master's thesis consists of six steps, described below:

First step is to investigate the dataset used for training, testing and validation in *Deep Long-Tailed Learning: A Survey*, as the class re-balancing in this paper is used as inspiration for this thesis.

Second step is generating a long-tailed version of CIFAR100 that can be used for training and comparisons of methods.

Third step is the implementation of models and methods, combining them.

Fourth step is hyperparameter settings.

Fifth step is training and evaluation of the models with different loss functions. The training is both on the balanced and long-tailed version of CIFAR100.

Sixth step is a comparison of the methods.
Other steps could involve comparisons to related work and other long-tailed learning methods.

1.1.4 Scope of this thesis

This thesis focuses on applying deep long-tailed learning methods to image classification tasks with an emphasis on loss functions as a solution to handle imbalanced datasets. The experiments are conducted on the CIFAR100 dataset, with both a balanced and synthetically generated long-tailed version. This thesis explores class re-balancing methods, specifically cost-sensitive learning applied via loss functions. These include cross-entropy loss, focal loss, weighted cross-entropy loss, class-balanced loss, balanced softmax loss, equalization loss, and LDAM loss. The evaluation is done with particular focus on top-1 accuracy, with attention paid to performance on head, middle, and tail classes. This thesis does not explore other long-tailed learning approaches such as re-sampling, information augmentation, or model architecture modifications.

1.2 Related Work

A section that describes the work related to this thesis. This includes re-sampling, data augmentation, module improvement.

Chapter 2

Background

This chapter presents the different background topics of the thesis work, which are the long-tailed datasets, model architectures *Convolutional Neural Networks (CNN)* and *Visual Transformers (VT)*, the deep long-tailed learning methods *Class Re-balancing (CR)*, *Information Augmentation (IA)*, and *Module Improvement (MI)*. These topics will be explained for the reader.

Mention image classification, as it is the primary goal of this thesis.

2.1 Long-Tailed Datasets

Long-tailed datasets pose significant challenges in deep learning, as they represent an extreme form of class imbalance. Addressing these challenges is central to this thesis, which explores methods to improve model performance on underrepresented classes. This section outlines the structure of long-tailed distributions and their implications.
A balanced dataset is one where all classes are evenly represented, whereas imbalanced datasets feature varying sample sizes across classes. Long-tailed datasets are characterized by a significant class imbalance, where a few dominant classes account for most samples (head classes), while the majority of classes are underrepresented (tail classes) as depicted in Figure 2.1. This distribution is common for real-world datasets [2, 3]. For example, the iNaturalist, a popular benchmark for image classification, exhibits a long-tailed distribution of species [4]. Other benchmarks are constructed by sampling from datasets such as ImageNet [5] and CIFAR-100 [6] using a Pareto distribution, which simulates long-tailed class distributions with a power-law decay [1, 7, 8].
One such benchmark, CIFAR100-LT [8], derived from the CIFAR-100 dataset [6], serves as the primary dataset for the experiments conducted in this thesis. CIFAR-100 is a widely used benchmark in classification research due to its diverse class representation and manageable size. It consists of 60,000 32x32 color images divided into 100 classes, each with 600 samples. These are further split into 500 training images and 100 testing images per class. CIFAR100-LT is created by reducing the number of samples in certain classes of CIFAR-100 following a

Pareto distribution with number of samplers per class as followed [9]:

$$\text{num\_samples} = \text{max\_samples} \times (\text{imb\_factor})^{\frac{\text{class\_index}}{\text{num\_classes}-1}} \tag{2.1}$$

where *max_samples* is the maximum number of samples for a class, *imb_factor* is the imbalance factor, *class_index* is the index of the class, and *num_classes* is the total number of classes.

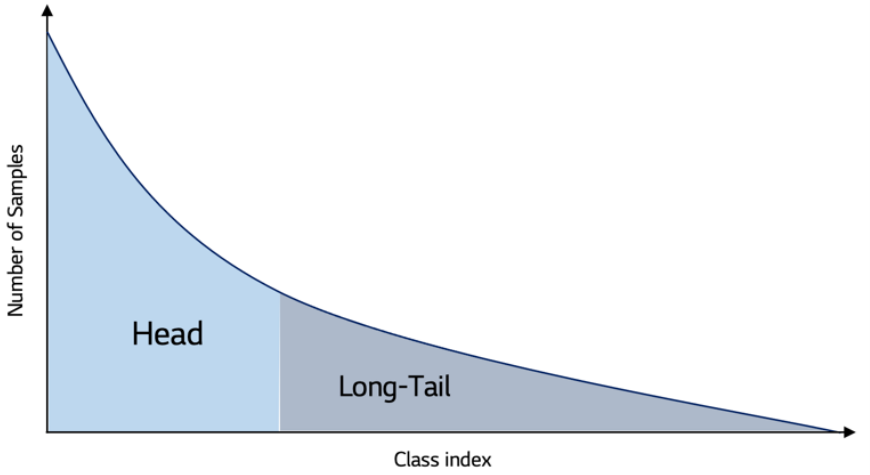

Figure 2.1: Illustration of a long-tailed distribution. Figure from [10].

Class imbalance has a profound impact on model performance compared to evenly distributed datasets [11, 12]. Deep networks trained on long-tailed datasets often exhibit biased performance, favoring head classes while performing poorly on tail classes [1]. Zhang et al. (2023) provide a comprehensive survey of methods addressing this challenge, categorizing current approaches into three main groups: class re-balancing, information augmentation, and module improvement. These methods will be further explored in section 2.3.

2.2 Model Architechtures

Deep learning has revolutionized image classification by introducing models capable of learning complex patterns and representations from data. Among these, Convolutional Neural Networks (CNNs) and Visual Transformers (VTs) stand out as the primary architectures used in this thesis. This section provides a theoretical foundation for these models, focusing on the specific architectures utilized: MobileNetV2 [13], ResNet50V2 [14], and ConvNeXt Base [15] as the CNN architectures, and ViT-B/16 [16] as the VT architecture.

2.2.1 Introdution to Deep Neural Networks

Before the introduction of Convolutional Neural Networks (CNNs) and, more recently, Vision Transformers (ViTs), the standard approach for image classification

involved flattening a two-dimensional image matrix into a one-dimensional array and passing it through a Multilayer Perceptron (MLP), also known as a feed-forward neural network. MLPs are fully connected neural networks composed of an input layer, output layer, and one or more hidden layers. Being fully connected means that each neuron in a given layer is connected to all neurons in the next layer, forming a dense network. These connections are associated with weights and biases, which the network learns during training. Input features are fed into the input layer, propagated through hidden layers that add complexity to model nonlinear relationships, and yield predictions in the output layer. Known as universal approximators, MLPs can approximate any continuous function given sufficient neurons in the hidden layers [17].
To illustrate the structure of a neural network, figure 2.2 shows an example of a feed-forward neural network with three input neurons, two hidden layers, each with four neurons, and two output neurons. This architecture could be used, for instance, to classify images of cats and dogs based on three input features, such as height, weight, and width of the animals. The input propagates through the network, with each neuron computing a weighted sum of its inputs followed by an optional nonlinearity. The final output is a prediction, where the class corresponding to the neuron with the highest value is chosen.

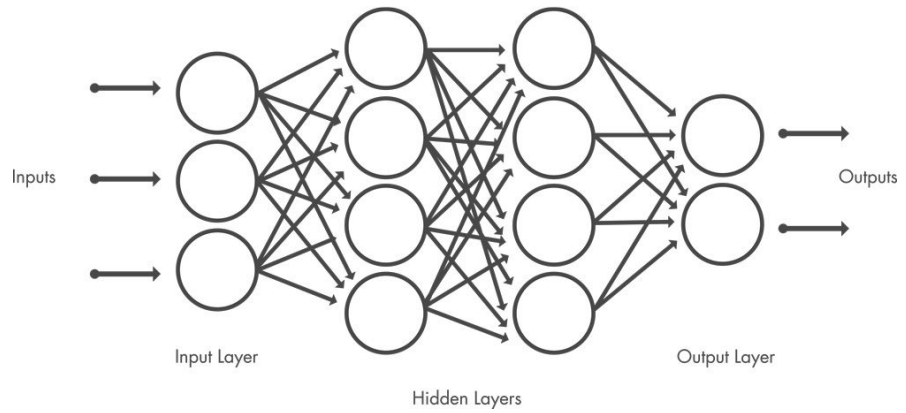

Figure 2.2: Layers of a convolutional neural network. Figure from [18].

However, this simple neural network becomes insufficient for more complex problems, such as image classification, as it requires an increasing number of parameters. This limitation was addressed by CNNs, which introduced convolutional and pooling layers to effectively preserve and utilize the spatial information of pixels in two-dimensional images [17].

2.2.2 Convolutional Neural Networks

A convolutional neural network is a neural network architecture for deep learning that gained popularity after being introduced by LeCun et al. in 1995 [19]. It is designed to recognize patterns in images for classification, as CNNs preserve the two dimensional input of an image, preserving the idea that nearby pixels are related [17].

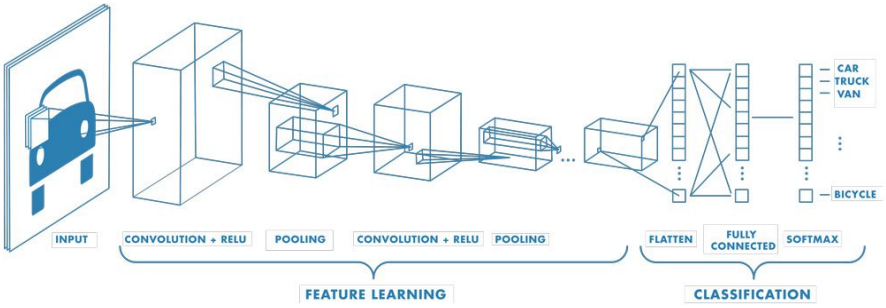

Figure 2.3: Illustration of a convolutional neural network. Figure from [18].

MobileNetV2 Architechture

MobileNetV2 is a lightweight CNN architecture designed for efficient mobile and edge computing applications [13]. It introduces two key innovations:
Inverted Residuals: Use of shortcut connections between thin bottleneck layers to improve performance while maintaining low computational cost. Linear Bottlenecks: Avoiding non-linearities in narrow layers to preserve information. MobileNetV2 is particularly suited for image classification tasks on resource-constrained devices and is used in this thesis for its efficiency and effectiveness in feature extraction.

ResNet50V2 Architechture

ResNet50V2 is a variant of the ResNet (Residual Network) architecture, which introduced the concept of residual learning to address the vanishing gradient problem in deep networks [14]. Key features include:
Residual Connections: Allow gradients to flow more effectively through deeper layers. Improved Training Dynamics: Incorporates pre-activation, leading to smoother optimization. ResNet50V2's ability to extract hierarchical features and its robustness to overfitting make it a popular choice for image classification tasks.

ConvNeXt Base Architechture

ConvNeXt Base is a modernized CNN architecture inspired by insights from transformer-based models, designed to achieve competitive performance while retaining the efficiency of CNNs [15]. Notable features include:
Simplified Design: Incorporates architectural refinements such as depthwise convolutions and LayerNorm. Enhanced Efficiency: Balances accuracy and computational cost, bridging the gap between traditional CNNs and transformer-based models. ConvNeXt Base represents a state-of-the-art approach to CNN design, making it a compelling choice for this thesis.

2.2.3 Visual Transformers

Explain their advantages over CNNs for certain tasks. Mention why they are relevant for handling long-tailed datasets.

ViT-B/16 Architecture

ViT-B/16 is a Visual Transformer model that leverages the transformer architecture for image classification [16]. Key characteristics include:
Patch Embeddings: Images are divided into 16x16 patches, which are then flattened and embedded. Self-Attention Mechanism: Captures global dependencies and relationships between image regions. Scalability: Performs particularly well on large-scale datasets, where its global attention mechanism can fully exploit the available data. ViT-B/16 exemplifies the application of transformers in computer vision and is used in this thesis to explore the capabilities of attention-based architectures.

2.3 Classic Long-Tailed Methods

Introduce the three methods (CR, IA, MI) with a brief explanation of their purpose.

Following the paper *Deep Long-Tailed Learning: A Survey* [1], the existing deep long-tailed learning methods are grouped into three main categories based on their technical approach: class re-balancing, information augmentation, and module improvement. These categories are further divided onto sub-categories: re-sampling, class-sensitive learning, logit adjustment, transfer learning, data augmentation, representation learning, classifier desing, decoupled training, and ensemble learning as shown on figure 2.4. This thesis does not aim to examine all the beforementioned method, but aims to find a deep learning approach to a specific problem. The backgrounds of the methods used in this thesis are described in this section.

2.3.1 Class Re-balancing

The class re-balancing method aims to re-balance the effect of the imbalanced training dataset, and has three main sub-categories: re-sampling, class-sensitive learning, and logit adjustment [1].

Re-sampling

The traditional way to sample when training deep networks is bases on mini-batch gradient descent with random sampling. This means that each sample has an equal probability of being sampled. When sampling from an imbalanced dataset, samples from head classes naturally occur more often, and thus have higher chance of being sampled than samples from tail classes, making the resulting deep models biased towards head classes. Re-sampling is a method that adresses this problem by adjusting the number of samples per class in each sample batch for model training.

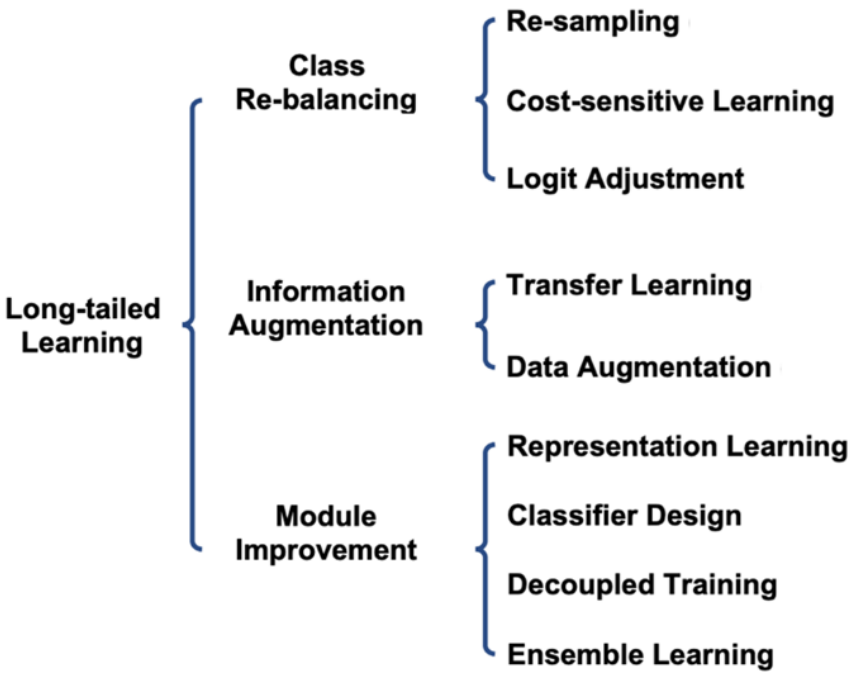

Figure 2.4: Long-tailed categories as described by *Zhang et al.* [1].

Class-sensitive Learning

Class-sensitive learning incorporates strategies to adjust the loss function, making it more sensitive to the imbalanced nature of the dataset. This approach directly modifies the optimization process to prioritize learning from under-represented tail classes.
TODO: Mention re-weighting and re-margining. Table/overview of loss functions as in the paper.

Loss Functions for Class-Sensitive Learning

The loss function serves as a measure of the model's fitness to the data, quantifying the distance between the actual and predicted values of the target. Typically, the loss is represented as a nonnegative value, where smaller values indicate a better fit, and a perfect fit corresponds to a loss of zero [17].
Conventional training of deep networks using the softmax cross-entropy loss often overlooks class imbalance. This results in uneven gradients for different classes, leading to suboptimal performance on underrepresented classes. To mitigate this issue, modifications to the loss function are introduced to ensure a more balanced contribution from each class during training. One such technique is re-weighting which adjusts the training loss for different classes by assigning a specific weight to each class [1]. The softmax cross-entropy loss is used as a baseline, and is described below along with the loss functions for re-weighting.

Softmax Cross-Entropy Loss The *Softmax-Cross-Entropy loss*, often referred to as *softmax loss*, is a widely used combination for training deep neural

networks in classification tasks, including image classification. It is particularly effective for multi-class problems, where the goal is to assign an input image to one of several predefined categories [20] [21].
The *Softmax* function transforms the raw output scores (logits) of the final layer of a neural network into a probability distribution over K classes. For an input $\mathbf{z} = [z_1, z_2, \ldots, z_K]$, the Softmax function for class i is defined as:

$$P(y = i \mid \mathbf{z}) = \frac{\exp(z_i)}{\sum_{j=1}^{K} \exp(z_j)} \tag{2.2}$$

Here, $\exp(z_i)$ ensures that all values are positive, and dividing by the sum normalizes the probabilities so that they sum to 1. This normalization is crucial for classification, as it allows the network's outputs to represent the likelihood of each class.
The *Cross-Entropy loss* measures the difference between the predicted probability distribution $\mathbf{P}$ (produced by Softmax) and the true distribution $\mathbf{y}$ (the one-hot encoded ground truth). It is defined as:

$$\mathcal{L}_{\text{CE}} = -\sum_{i=1}^{K} y_i \log(P(y = i \mid \mathbf{z})) \tag{2.3}$$

For a single example where the true class is c, this simplifies to:

$$\mathcal{L}_{\text{CE}} = -\log(P(y = c \mid \mathbf{z})) \tag{2.4}$$

This formulation penalizes incorrect predictions by heavily weighting the log of the predicted probability for the true class. The loss is minimized when the predicted probability $P(y = c \mid \mathbf{z})$ approaches 1, indicating high confidence in the correct class.
This combination has become the de facto standard for image classification tasks, providing a robust and mathematically sound framework for training deep neural networks.

Weighted Softmax Cross-Entropy Loss The *Weighted Softmax Cross-Entropy loss*, often referred to as *weighted softmax loss*, is a variant of the standard softmax cross-entropy loss, designed to address imbalanced datasets [21] [22]. By assigning different weights to each class, this method ensures that underrepresented classes contribute more to the overall loss, improving the model's performance on minority classes. The weighted cross-entropy loss applies class-specific weights to the standard cross-entropy formulation. It is defined as:

$$\mathcal{L}_{\text{WCE}} = -\sum_{i=1}^{K} w_i y_i \log(P(y = i \mid \mathbf{z})) \tag{2.5}$$

Where w_i is the weight for class i, reflecting its relative importance, y_i is the one-hot encoded true label for class i, and $P(y = i \mid \mathbf{z})$ is the predicted probability for class i.
For a single example where the true class is c, the loss simplifies to:

$$\mathcal{L}_{\text{WCE}} = -w_c \log(P(y = c \mid \mathbf{z})) \tag{2.6}$$

This weighted formulation ensures that minority classes contribute more to the overall loss, addressing the imbalance during training and improving the model's performance on underrepresented classes.

Focal Loss Focal Loss, introduced by Lin et al. (2017) [22], addresses the challenges of extreme class imbalance in classification tasks by dynamically scaling the standard cross-entropy loss. Focal Loss mitigates the issue of imbalanced datasets by down-weighting the loss contributions from well-classified examples and focusing on misclassified examples during training.

Class-Balanced Loss *Class-balanced loss*, introduced by Cui et al. (2019) [12], ...

Balanced Softmax Loss *Balanced Softmax loss*, introduced by Ren et al. (2020) [23], ...

LDAM Loss *LDAM loss*, introduced by Cao et al. (2019) [8], ...

Equalization Loss *Equalization loss*, introduced by Tan et al. (2020) [24], ...

2.3.2 Information Augmentation

Data augmentation techniques tailored for long-tailed datasets.

Transfer Learning

Data Augmentation

2.3.3 Module Improvement

Architectural changes to improve tail-class representation.

Chapter 3

Methodology

This chapter describes the methods and approaches used in the experiments. This includes dataset preparation, models, loss functions, etc.

3.1 Overview of Approach

An overall description of the approach to tackling the long-tailed dataset problem, including an explanation of the strategy, such as balancing techniques and model selection.

3.2 Dataset Preparation and Specifications

A description of this section here.
Following the dataset structure used in *Deep Long-Tailed Learning: A Survey*, the CIFAR100 dataset was modified to create a long-tailed training set and a balanced test set. Key details include:

- Dataset characteristics: Number of classes, imbalance ratio, and train-test splits.
- Preprocessing steps: Resizing, normalization, and augmentation techniques.
- Handling imbalance: Techniques like re-sampling and augmentation to address the long-tailed distribution.

3.2.1 Data Characteristics: Class Distribution

A list of subjects to include in this section:

- Describe the ImageNet-LT dataset: number of classes, imbalance ratio, etc.
- Describe the plots and what they mean for the CIFAR100-LT data preparation.

The first step is to prepare the data for training and testing. In order to generate training, validation, and test datasets that resemble the datasets used for the empirical studies in *Deep Long-Tailed Learning: A Survey* [1], their dataset are investigated: The GitHub repository [25] for the paper *Deep Long-Tailed Learning: A Survey* was downloaded and an environment was set up on the Jupyter Hub on the Freja node on the ece cluster. The `.txt` files with the data from ImageNet-LT (`ImageNet_LT_train.txt`, `ImageNet_LT_val.txt`, `ImageNet_LT_test.txt`) are shown on figures 3.1, 3.2, and 3.3, respectively.

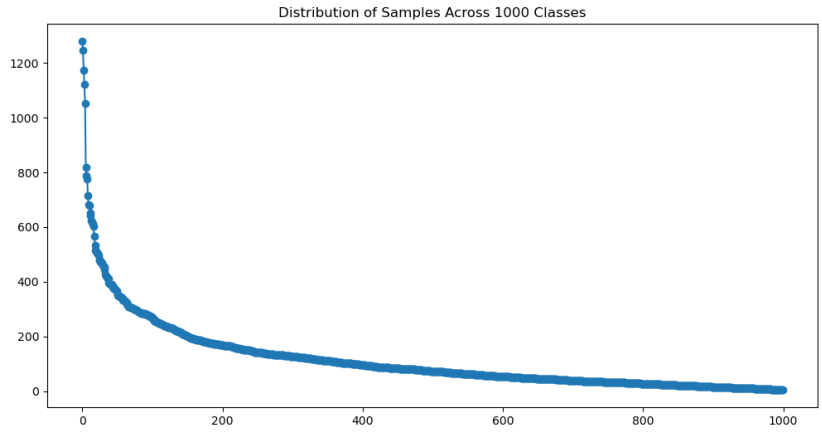

Figure 3.1: The class distribution of the training images for the ImageNet-LT dataset shows a long-tailed distribution.

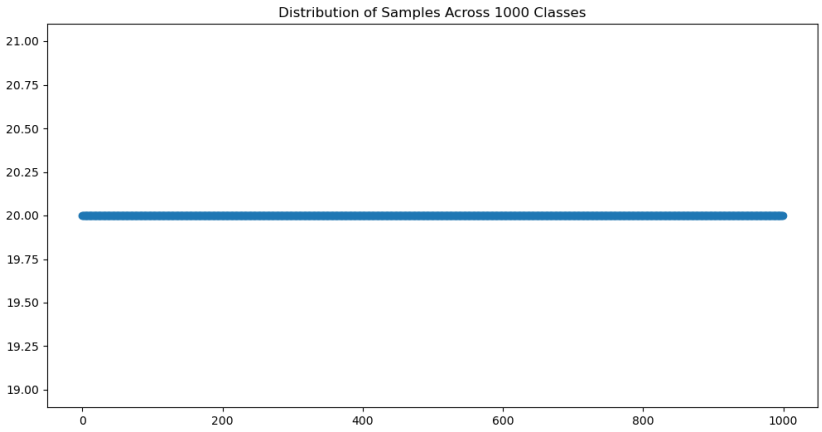

Figure 3.2: The class distribution of the validation images for the ImageNet-LT dataset shows that there are 20 samples of each class.

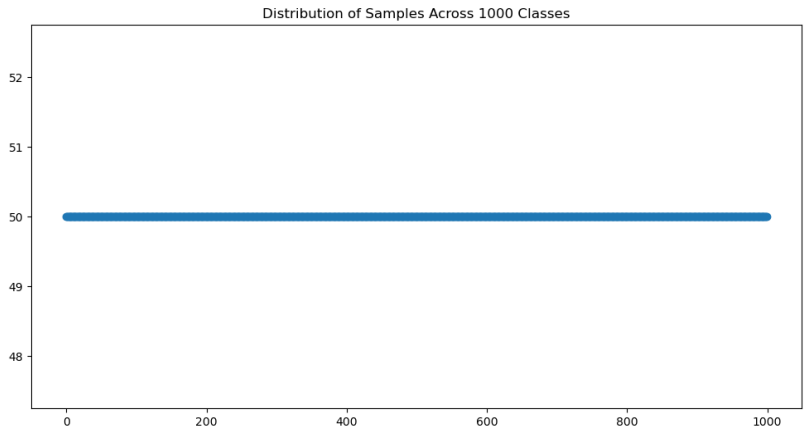

Figure 3.3: The class distribution of the test images for the ImageNet-LT dataset shows that there are 50 samples of each class.

3.2.2 Preparation: CIFAR100-LT

A list of subjects to include in this section:

- Briefly describe the CIFAR100 dataset, and why it was chosen as the primary dataset for this thesis. Refer to section 2.1.
- Insert plots of the CIFAR100 dataset.
- Describe the generation of the imbalanced dataset: IMBALANCECIFAR100 method from the LDAM-DRW paper.
- Describe the imbalance ratio.
- Explain why the dataset was saved, and not generated in run-time, like in LDAM-DRW.
- Explain why the dataset was split into 450 samlpes per class for training and 50 samples per class for testing.
- Insert plots of the imbalanced dataset.
- Describe the head, middle, and tail classes.
- Insert plot of the division of the long-tailed test dataset: head, middle, and tail classes.
- Explain the purpose of the division.
- Potentially a few images from the CIFAR100 dataset.
- Argument as to why the training data was split into training and test.

The experiments conducted in this thesis primarily utilize the CIFAR-100 dataset.

The dataset was downloaded using the PyTorch torchvision.datasets.CIFAR100 utility. The training and test datasets were preprocessed by converting the images to tensors using the ToTensor transformation and saved as .pth files for efficient loading during experiments.

To address the needs of the experiments in this thesis, the original CIFAR-100 dataset was modified to create a new split of the training data. Specifically, the training data was split into 450 samples per class for training and 50 samples per class for testing, maintaining the class distribution within these splits. The original test set of the CIFAR-100 dataset was retained as the validation set for training and evaluation purposes.

Training Set: To simulate real-world scenarios with class imbalances, the training dataset was modified to introduce an exponential imbalance across the 100 classes. The imbalance was created using the quantile Pareto distribution in equation (2.1), where the number of samples per class decreases exponentially, controlled by the imbalance factor. For this thesis, an imbalance factor of 0.01 was applied. This means that the most frequent class contains significantly more samples than the least frequent class.

The resulting class distribution varied from the most frequent class having 450 samples to the least frequent class having only [todo: check] samples. This imbalance ensured no class was left with zero samples, maintaining the integrity of all classes for training. TODO: see figure 3.4.

Test Set: To evaluate the performance of the model under similar conditions to the imbalanced training set, an imbalanced test set was created from the previously split test dataset. The imbalance in the test set mirrors the exponential distribution used for the training data, with the same imbalance factor of 0.01. The class distribution in the test set follows the same order of classes (from most to least frequent) as the imbalanced training set. No class has fewer than one sample.

Describe the class distribution in figures 3.4 and 3.5.

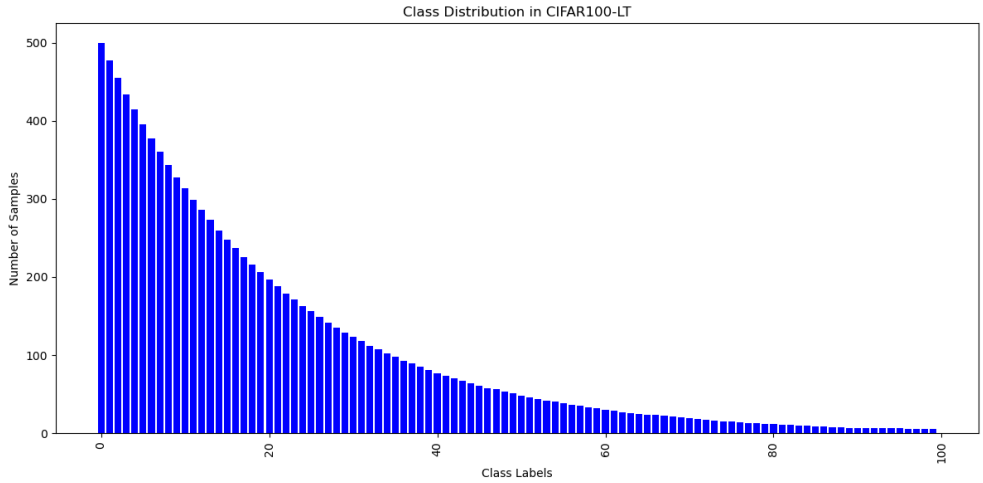

Figure 3.4: The class distribution of CIFAR100-LT with imbalance ratio 100 generated by the imbalance_cifar.py in the LDAM-DRW GitHub repository [9].

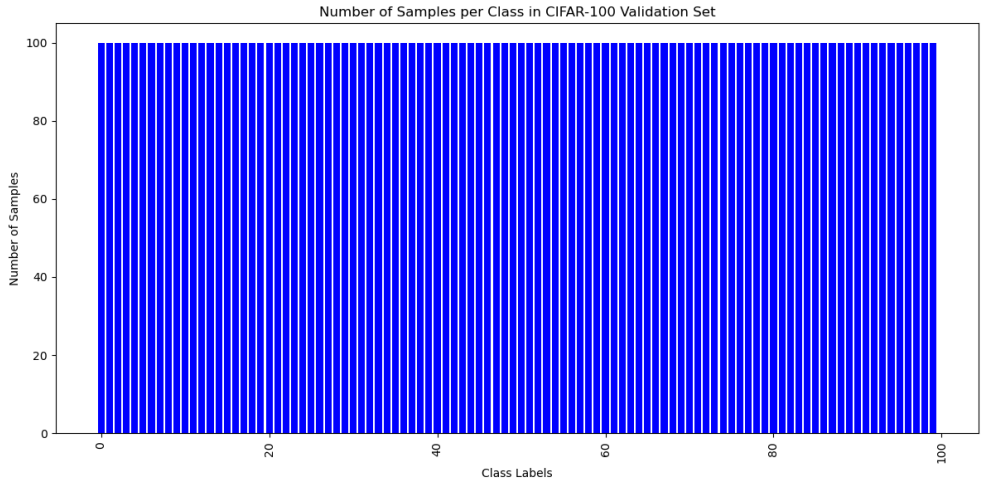

Figure 3.5: The class distribution of the CIFAR100 validation set from torchvision [reference here].

3.2.3 Data Augmentation

Not sure if this should be a section, or if it should be in the experimental setup section.
Describe what data augmentation was used on the training data.

3.3 Long-tailed Learning Techniques

Description of the specific methods used to address class imbalance, such as data sampling, class re-weighting, etc. Justification for selecting these techniques, po-

tentially referencing prior research (from Deep Long-Tailed Learning: A Survey by Zhang et al.).

3.3.1 Model Selection

A list of subjects to include in this section:

- Mention the model architectures chosen for training, and describe why they are appropriate for deep long-tailed learning with reference to the background section.
- Discuss the strengths and limitations of these models in addressing the challenges posed by imbalanced data.
- Describe how they were pretrained (ImageNet-1K, ImageNet-21K) and what that means for the training on CIFAR100.

3.3.2 Selection of Loss Function

A list of subjects to include in this section:

- Describe the different loss functions and why they are appropriate for deep long-tailed learning with reference to the background section.
- Rationale for each loss function's inclusion, focusing on its expected benefits for imbalanced classes and how it adresses the bias toward majority classes.

3.3.3 Excluded Long-Tailed Learning Methods

This section will explain why some deep long-tailed learning techniques, like resampling, was not the focus of this thesis.

3.4 Evaluation Metrics

A list of subjects to include in this section:

- Describe common evaluation metrics used for classification tasks.
- Explain the choice of top-1 accuracy.
- Explain how the performance is assessed across different class groups.
- Explain the choice of F1-score.

3.5 Reproducibility

A list of subjects to include in this section:

- Use of random seed initialization.
- Documentation of dataset versions and codebase.
- Availability of scripts for dataset preparation and model training.

3.6 Implementation Details

Technical explanations of any unique or customized methods implemented in code, for example the custom dataset.
A list of subjects to include in this section:

- Explain how the models where implemented.
- Rationale for implementing the loss functions manually instead of copy existing repositories.
- Describe the benefits of copying existing repositories.

Chapter 4

Experimental Setup

This chapter focuses on the on the implementation details of the experiments conducted in this thesis. Here, the specifics of the training configurations are described.

4.1 Dataset Specifications

Table 4.1 provides an overview of the dataset splits used in this thesis. The training set consists of 45,000 samples, with 450 samples per class, generated by splitting the original CIFAR-100 training set. The validation set, unaltered from the original CIFAR-100 test set, contains 10,000 samples with an equal distribution of 100 samples per class. The test set was derived from the split training data and consists of 5,000 samples, with 50 samples per class.
To simulate real-world class imbalance scenarios, an exponential imbalance was introduced into the training and test sets. The imbalance factor (imb_factor) was set to 0.01, resulting in a significant reduction of samples for the least frequent classes. Table 4.2 provides a summary of the imbalance characteristics.
Additionally, the imbalanced test set was further divided into three subsets based on class frequencies in the training data:

- **Head Test Set:** Includes the top one-third most frequent classes.
- **Middle Test Set:** Includes the middle one-third of classes.
- **Tail Test Set:** Includes the bottom one-third least frequent classes.

The class splits were determined by sorting classes based on the number of samples in the imbalanced training dataset and grouping them equally into three categories. The resulting test splits ensure that the model is evaluated on head, middle, and tail subsets, highlighting performance variations across different class frequencies.

Table 4.1: Dataset Specifications

Dataset Component	Total Samples	Samples per Class
Training Set	45,000	450
Validation Set	10,000	100
Test Set	5,000	50
Head Test Set	TODO	Variable
Middle Test Set	TODO	Variable
Tail Test Set	TODO	Variable

Table 4.2: Imbalance Specifications

Aspect	Details
Imbalance Type	Exponential
Imbalance Factor	0.01
Training Set Distribution	Most frequent class: 450 samples Least frequent class: 4 samples
Test Set Distribution	Mirrors training distribution Ensures no class has fewer than 1 sample

4.2 Data Preprocessing

Table 4.3 summarizes the preprocessing steps applied to the training, validation, and test datasets. [TODO: reference CIFAR100 statistics.]

Table 4.3: Data Preprocessing Steps

Dataset	Preprocessing Steps
Training	• Resize to 224×224 pixels • Random crop to 224×224 with 4 pixels of padding • Random horizontal flip • Normalize using CIFAR-100 statistics: Mean = [0.4914, 0.4822, 0.4465], Std = [0.2023, 0.1994, 0.2010]
Validation/Test	• Resize to 224×224 pixels • Normalize using CIFAR-100 statistics: Mean = [0.4914, 0.4822, 0.4465], Std = [0.2023, 0.1994, 0.2010]

4.3 Model Architecture Settings

Four different architectures were used: MobileNetV2, ResNet50V2, ViT-B/16, and ConvNeXt Base. The specifications of the model architectures can be seen in table 4.4.

Table 4.4: Model Architecture Settings

Model Name	Pretrained Weights	Modifications for CIFAR-100
MobileNetV2	`MobileNet_V2_Weights.IMAGENET1K_V1`	Replaced classification layer with a 100-class fully connected layer
ResNet50V2	`ResNet50_Weights.IMAGENET1K_V2`	Replaced the `fc` layer with a 100-class fully connected layer
ViT-B/16	`timm vit_base_patch16_224` pretrained	Replaced the `head` with a 100-class fully connected layer
ConvNeXt Base	`ConvNeXt_Base_Weights.DEFAULT`	Replaced the final layer of the classifier with a 100-class fully connected layer

All models were initialized with pretrained weights from ImageNet to leverage transfer learning. The modifications ensured that the architectures were adapted to the CIFAR-100 dataset while retaining the general features learned during pretraining.

4.4 Training Configurations

This section outlines the key hyperparameters and settings used during the training and evaluation of the models.

- **Optimizer**: Adam
- **Learning Rate**: Configured through the `learning_rate` parameter in the configuration file, with an initial value of `0.001`.
- **Batch Size**: Configured through the `batch_size` parameter, with a default value of `128`.
- **Number of Epochs**: Configured through the `num_epochs` parameter, with a default value of `90`.
- **Learning Rate Scheduler**: StepLR with a step size of `30` epochs and a decay factor of `0.1`.
- **Regularization Techniques**:
  - Weight Decay: Configured through the `weight_decay` parameter, with a default value of `1e-4`.
  - Early Stopping: Monitored validation loss with a patience value of `5` epochs.
- **Early Stopping Criteria**: The best model is saved based on the highest Top-1 validation accuracy.

- **Class Weights**: Dynamically computed based on the training dataset and passed to the loss function.
- **Device Setup**: Utilized `torch.nn.DataParallel` for multi-GPU training, with the primary device configured through the `device` parameter (`cuda` or `cpu`).

4.5 Evaluation Metrics

The following metrics were used to evaluate model performance during validation and testing:

- **Top-1 Accuracy**: Proportion of correct predictions across all classes.
- **F1 Score**:
  - **Macro F1**: Used for balanced datasets, e.g., validation set and balanced test set.
  - **Weighted F1**: Used for imbalanced datasets to account for class frequency differences.
- **Head/Middle/Tail Class Evaluation**:
  - The test set was divided into three subsets: head, middle, and tail classes, based on class frequency in the training dataset.
  - Performance on each subset was evaluated using both:
    - **Top-1 Accuracy**
    - **Weighted F1 Score**

4.6 Hardware and Software Configurations

The experiments were conducted on a high-performance computing system with the following hardware and software configurations:

4.6.1 Hardware

- **GPUs**: 4 NVIDIA TITAN X (Pascal), each with 12 GB memory.
- **RAM**: 125 GiB
- **Swap Space**: 63 GiB
- **CUDA Version**: 12.4
- **Driver Version**: 550.90.07

4.6.2 Software

- **Operating System**: Ubuntu 22.04.4 LTS (Jammy Jellyfish)
- **Python Version**: 3.11.8
- **Deep Learning Frameworks and Libraries Versions**:
  - PyTorch ($\geq$1.7)
  - Torchvision ($\geq$0.8.0)
  - Tensorboard ($\geq$1.14)

4.7 Reproducibility Considerations

To ensure that the experiments conducted in this thesis are reproducible, the following measures were implemented:

- **Random Seed**:
  - A fixed random seed of `42` was used for all experiments to ensure consistent initialization across runs.
  - Randomness was controlled for:
    - Python's `random` library.
    - NumPy (`np.random.seed`).
    - PyTorch (`torch.manual_seed` and `torch.cuda.manual_seed`).
  - `cudnn.deterministic` was set to `True` to enforce deterministic behavior in GPU computations.
- **Configuration Management**:
  - All hyperparameters, dataset settings, and model configurations were defined in YAML configuration files.
  - This allows for the exact replication of experiments by simply reusing the configuration files.
- **Saved Artifacts**:
  - Trained models and datasets were saved in organized directories, making them accessible for evaluation or reuse in future experiments.
  - Model checkpoints were saved after achieving the best validation accuracy.
- **Environment Details**:
  - All library versions, operating system details, and hardware configurations were documented to ensure reproducibility.
  - The complete list of Python dependencies is provided in the project's documentation.

Chapter 5

Results and Analysis

Presentation of the results, with tables, charts, and explanations for each tested method's performance.
Brief overview of the chapter's purpose. Recap the evaluation goals (model performance across head, middle, and tail classes, and comparing methods).

5.1 Overall Results

Present the performance of all tested models and methods. Use tables or charts to summarize key results. Highlight trends or notable observations across the methods.

5.1.1 MobileNetV2

Table 5.1 show the top 1 accuracies for MobileNetV2 on various loss functions.

Loss Function	Balanced	Long-tailed	Head	Middle	Tail
Softmax	0.7978	0.8059	0.8069	0.7870	0.8684
Focal loss	0.8014	0.8011	0.7998 4	0.7870	0.8947
Weighted Softmax loss	0.7978	0.8059	0.8069	0.7870	0.8684
Class-balanced loss	0.7978	0.8059	0.8069	0.7870	0.8684
Balanced Softmax loss	0.8034	0.8030	0.8069	0.7574	0.9211
Equalization loss	0.7994	0.8040	0.8057	0.7692	0.9211
LDAM loss	0.7828	0.7821	0.7808	0.7574	0.9211

Table 5.1: Evaluation results for MobileNetV2 trained on the custom balanced dataset, showing Acc1.

Table 5.2 show the top 1 accuracies for MobileNetV2 on various loss functions.

Loss Function	Balanced	Long-tailed	Head	Middle	Tail
Softmax	0.5282	0.7735	0.8341	0.5917	0.2368
Focal loss	0.5200	0.7745	0.8389	0.5917	0.1579
Weighted Softmax loss	0.5016	0.7231	0.7808	0.5503	0.2105
Class-balanced loss	0.1936	0.0913	0.0521	0.2485	0.2632
Balanced Softmax loss	0.5796	0.7650	0.8069	0.6331	0.4211
Equalization loss	0.5310	0.7650	0.8235	0.5917	0.2368
LDAM loss	0.4264	0.5899	0.6137	0.5444	0.2632

Table 5.2: Evaluation results for MobileNetV2 trained on the long-tailed dataset showing Acc1.

Comparison

The comparison of my experiment to the original MobileNetV2 trained on the original CIFAR100 dataset.

Aspect	**Your Experiment**	**Their Study (A3)**
Dataset	CIFAR100 Customized	CIFAR100
Loss Function	Softmax Cross-Entropy	Binary Cross-Entropy
Epochs	90	100
Optimizer	Adam	LAMB
Learning Rate	Step decay at 30, 60 epochs	Cosine decay from 0.005 or 0.008
Augmentation	Resize (224x224), RandomCrop, Horizontal Flip, Normalize	RandAugment, Mixup, CutMix, Normalize
Hardware	4x NVIDIA TITAN X (Pascal, 12GB)	4x NVIDIA V100 (32GB)
Evaluation Metric	Top-1 Accuracy	Top-1 Accuracy
Top-1 Accuracy	79.8 %	86.2%-86.9%

Table 5.3: Comparison of MobileNetV2 on CIFAR100 with their study (Procedure A3) [26].

5.1.2 ResNet50V2

Table 5.4 show the top 1 accuracies for ResNet50V2 on various loss functions.

Loss Function	Balanced	Long-tailed	Head	Middle	Tail
Softmax loss	0.8324	0.8421	0.8448	0.8047	0.9474
Focal loss	0.8310	0.8344	0.8341	0.8166	0.9211
Weighted Softmax loss	0.8324	0.8421	0.8448	0.8047	0.9474
Class-balanced loss	0.8324	0.8421	0.8448	0.8047	0.9474
Balanced Softmax loss	0.8310	0.8430	0.8460	0.8107	0.9211
Equalization loss	0.8292	0.8373	0.8412	0.7929	0.9474
LDAM loss	0.7990	0.7983	0.8069	0.7337	0.8947

Table 5.4: Evaluation results for ResNet50V2 trained on the custom balanced dataset, showing Acc1.

Table 5.5 show the top 1 accuracies for ResNet50V2 on various loss functions.

Loss Function	Balanced	Long-tailed	Head	Middle	Tail
Softmax loss	0.5522	0.7954	0.8531	0.6391	0.2105
Focal loss	0.5456	0.7935	0.8483	0.6272	0.3158
Weighted Softmax loss	0.4976	0.7336	0.7915	0.5562	0.2368
Class-balanced loss	0.2052	0.1836	0.1445	0.3787	0.1842
Balanced Softmax loss	0.5908	0.7916	0.8270	0.6568	0.6053
Equalization loss	0.5452	0.7897	0.8389	0.6450	0.3421
LDAM loss	0.3742	0.5937	0.6469	0.4438	0.0789

Table 5.5: Evaluation results for ResNet50V2 trained on the long-tailed dataset, showing Acc1.

Comparison

5.1.3 ViT-B/16

Table 5.6 show the top 1 accuracies for ViT-B/16 on various loss functions.

Loss Function	Balanced	Long-tailed	Head	Middle	Tail
Softmax loss	0.5620	0.5671	0.5521	0.6036	0.7368
Focal loss	0.5516	0.5538	0.5438	0.5680	0.7105
Weighted Softmax loss	0.5620	0.5671	0.5521	0.6036	0.7368
Class-balanced loss	0.5620	0.5671	0.5521	0.6036	0.7368
Balanced Softmax loss	0.5628	0.5642	0.5640	0.5325	0.7105
Equalization loss	0.5634	0.5519	0.5462	0.5503	0.6842
LDAM loss	0.5906	0.6013	0.5924	0.6095	0.7632

Table 5.6: Evaluation results for ViT-B/16 trained on the custom balanced dataset, showing Acc1.

Table 5.7 show the top 1 accuracies for ViT-B/16 on various loss functions.

Loss Function	Balanced	Long-tailed	Head	Middle	Tail
Softmax loss	0.2254	0.4367	0.5071	0.1775	0.0263
Focal loss	0.2210	0.4206	0.4834	0.1953	0.0263
Weighted Softmax loss	0.1284	0.1760	0.1919	0.1302	0.0263
Class-balanced loss	0.0558	0.0076	0.0000	0.0237	0.1053
Balanced Softmax loss	0.2460	0.4244	0.4822	0.2130	0.0789
Equalization loss	0.2168	0.4215	0.4893	0.1716	0.0263
LDAM loss	0.5906	0.6013	0.5924	0.6095	0.7632

Table 5.7: Evaluation results for ViT-B/16 trained on the long-tailed dataset, showing Acc1.

Comparison

Aspect	**Your Experiment**	**PUGD Results**
Dataset	CIFAR100	CIFAR100
Model	ViT-B/16 pretrained on ImageNet-21K	Multiple models: VGG-16, ResNet-18, DenseNet-121, UPANet-16, and ViT-B/16
Pretraining	ImageNet-21K	ImageNet-1K (for fine-tuned models)
Optimizer	Adam	PUGD
Loss Function	Softmax Cross-Entropy	Softmax Cross-Entropy
Epochs	90	200 (end-to-end); 80–100 (fine-tuning)
Learning Rate	Step decay: 0.001 → 0.0001 → 0.00001	Cosine Annealing: 0.1 (end-to-end); 0.01–0.005 (fine-tuning)
Augmentation	Resize (224), RandomCrop, Horizontal Flip, Normalize	Resize (224), RandAugment, Cutout, Normalize
Top-1 Accuracy	-	End-to-End: Up to 78.30% (DenseNet-121); Fine-tuning: ViT-B/16 achieves 93.95%
Hardware	4x NVIDIA TITAN X (Pascal, 12GB)	RTX-Titan, 32GB RAM, eight-core processor

Table 5.8: Comparison of my experiment with PUGD results on CIFAR-100.

5.1.4 ConvNeXt Base

Table 5.9 show the top 1 accuracies for ConvNeXt Base on various loss functions.

Loss Function	Balanced	Long-tailed	Head	Middle	Tail
Softmax loss	0.8332	0.8535	0.8566	0.8166	0.9474
Focal loss	0.8314	0.8487	0.8507	0.8284	0.8947
Weighted Softmax loss	0.8332	0.8535	0.8566	0.8166	0.9474
Class-balanced loss	0.8332	0.8535	0.8566	0.8166	0.9474
Balanced Softmax loss	0.8364	0.8344	0.8365	0.7988	0.9474
Equalization loss	0.8318	0.8468	0.8448	0.8343	0.9474
LDAM loss	0.8316	0.8373	0.8412	0.8047	0.8947

Table 5.9: Evaluation results for ConvNeXt Base trained on the custom balanced dataset, showing Acc1.

Table 5.10 show the top 1 accuracies for ConvNeXt Base on various loss functions.

Loss Function	Balanced	Long-tailed	Head	Middle	Tail
Softmax loss	0.5972	0.8316	0.8898	0.6568	0.3158
Focal loss	0.5938	0.8145	0.8685	0.6568	0.3158
Weighted Softmax loss	0.4090	0.6356	0.6848	0.4911	0.1842
Class-balanced loss	0.0142	0.0019	0.0000	0.0000	0.0526
Balanced Softmax loss	0.6460	0.8230	0.8685	0.6509	0.5789
Equalization loss	0.5956	0.8278	0.8768	0.6923	0.3421
LDAM loss	0.3770	0.5956	0.6445	0.4260	0.2632

Table 5.10: Evaluation results for ConvNeXt Basetrained on the long-tailed dataset, showing Acc1.

Comparison

5.2 Head, Middle, and Tail Class Performance

Break down the performance into head, middle, and tail class groups. Include visualizations. Discuss how well the methods balance performance across these groups, particularly focusing on tail classes.

5.3 Comparison with Baselines

5.4 Comparison of Loss Functions

Analyze how different loss functions impact performance. Use visualizations to compare results (per-class performance or confusion matrices). Discuss strengths and weaknesses of each loss function.

5.5 Qualitative Results

Optional.
Provide examples of correctly and incorrectly classified samples, especially for tail classes. Include visualizations or images of difficult cases to highlight challenges in tail-class prediction.

5.6 Summary and Discussion

Recap the key findings, such as which methods or loss functions performed best and why. Connect these findings to the thesis objectives and broader implications for long-tailed learning.

Chapter 6

Conclusion and Future Work

Summary of the work, contributions, and suggestions for future improvements or research directions.

6.1 Revisiting the Goals of the Thesis

6.2 Future Work

Bibliography

[1] Yifan Zhang et al. "Deep long-tailed learning: A survey". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2023).

[2] MEJ Newman. "Power laws, Pareto distributions and Zipf's law". In: *Contemporary Physics* 46.5 (Sept. 2005), pp. 323–351. ISSN: 1366-5812. DOI: 10.1080/00107510500052444. URL: http://dx.doi.org/10.1080/00107510500052444.

[3] Ziwei Liu et al. *Large-Scale Long-Tailed Recognition in an Open World*. 2019. arXiv: 1904.05160 [cs.CV]. URL: https://arxiv.org/abs/1904.05160.

[4] Grant Van Horn et al. *The iNaturalist Species Classification and Detection Dataset*. 2018. arXiv: 1707.06642 [cs.CV]. URL: https://arxiv.org/abs/1707.06642.

[5] Jia Deng et al. "ImageNet: A large-scale hierarchical image database". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848.

[6] Alex Krizhevsky and Geoffrey Hinton. *Learning Multiple Layers of Features from Tiny Images*. Technical Report. University of Toronto, 2009.

[7] Charika de Alvis and Suranga Seneviratne. *A Survey of Deep Long-Tail Classification Advancements*. 2024. arXiv: 2404.15593 [cs.LG]. URL: https://arxiv.org/abs/2404.15593.

[8] Kaidi Cao et al. *Learning Imbalanced Datasets with Label-Distribution-Aware Margin Loss*. 2019. arXiv: 1906.07413 [cs.LG]. URL: https://arxiv.org/abs/1906.07413.

[9] Di Kai. *LDAM-DRW: Learning from Long-Tailed Data*. GitHub repository, [Online]. Available: https://github.com/kaidic/LDAM-DRW/tree/master. Accessed: 2024-09-18.

[10] LG AI Research. *[ICML 2022] Part 1: Long-Tail Distribution Learning*. https://www.lgresearch.ai/blog/view/?seq=257&page=1&pageSize=12. Accessed: 2024-11-26.

[11] Grant Van Horn and Pietro Perona. *The Devil is in the Tails: Fine-grained Classification in the Wild*. 2017. arXiv: 1709.01450 [cs.CV]. URL: https://arxiv.org/abs/1709.01450.

[12] Yin Cui et al. *Class-Balanced Loss Based on Effective Number of Samples*. 2019. arXiv: 1901.05555 [cs.CV]. URL: https://arxiv.org/abs/1901.05555.

[13] Mark Sandler et al. "Mobilenetv2: Inverted residuals and linear bottlenecks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 4510–4520.

[14] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90.

[15] Agastya Todi et al. "ConvNext: A Contemporary Architecture for Convolutional Neural Networks for Image Classification". In: *2023 3rd International Conference on Innovative Sustainable Computational Technologies (CISCT)*. IEEE. 2023, pp. 1–6.

[16] Alexey Dosovitskiy et al. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. 2021. arXiv: 2010.11929 [cs.CV]. URL: https://arxiv.org/abs/2010.11929.

[17] Aston Zhang et al. *Dive into Deep Learning*. https://D2L.ai. Cambridge University Press, 2023.

[18] *What Is a Convolutional Neural Network?* MathWorks, [Online]. Available: https://www.mathworks.com/discovery/convolutional-neural-network.html. Accessed: 28-Nov-2024.

[19] Yann Lecun et al. "Comparison of learning algorithms for handwritten digit recognition". In: Jan. 1995.

[20] X. L. Chaitanya Asawa. *CS231n: Convolutional Neural Networks for Visual Recognition*. http://cs231n.github.io/. Stanford, [Online]. 2024.

[21] *torch.nn.CrossEntropyLoss — PyTorch documentation*. https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html. Accessed: 2024-11-20.

[22] Tsung-Yi Lin et al. *Focal Loss for Dense Object Detection*. 2018. arXiv: 1708.02002 [cs.CV]. URL: https://arxiv.org/abs/1708.02002.

[23] Jiawei Ren et al. *Balanced Meta-Softmax for Long-Tailed Visual Recognition*. 2020. arXiv: 2007.10740 [cs.LG]. URL: https://arxiv.org/abs/2007.10740.

[24] Jingru Tan et al. *Equalization Loss for Long-Tailed Object Recognition*. 2020. arXiv: 2003.05176 [cs.CV]. URL: https://arxiv.org/abs/2003.05176.

[25] Vanint. *Awesome-LongTailed-Learning*. https://github.com/Vanint/Awesome-LongTailed-Learning. Accessed: 2024-09-18. 2023.

[26] Ross Wightman, Hugo Touvron, and Hervé Jégou. *ResNet strikes back: An improved training procedure in timm*. 2021. arXiv: 2110.00476 [cs.CV]. URL: https://arxiv.org/abs/2110.00476.

Appendix A

Results

Tables of the results from training.

A.1 MobileNetV2

MobileNetV2 trained on custom balanced dataset and imbalanced dataset on different loss functions.

Table A.1 show the top 1 accuracies for MobileNetV2 on various loss functions.
Table A.2 show the loss, top 1 accuracy, and F1 score.
Table A.3 show the top 1 accuracies for MobileNetV2 on various loss functions.
Table A.4 show the loss, top 1 accuracy, and F1 score.

Loss Function	Balanced	Long-tailed	Head	Middle	Tail
Softmax	0.7978	0.8059	0.8069	0.7870	0.8684
Focal loss	0.8014	0.8011	0.7998 4	0.7870	0.8947
Weighted Softmax loss	0.7978	0.8059	0.8069	0.7870	0.8684
Class-balanced loss	0.7978	0.8059	0.8069	0.7870	0.8684
Balanced Softmax loss	0.8034	0.8030	0.8069	0.7574	0.9211
Equalization loss	0.7994	0.8040	0.8057	0.7692	0.9211
LDAM loss	0.7828	0.7821	0.7808	0.7574	0.9211

Table A.1: Evaluation results for MobileNetV2 trained on the custom balanced dataset, showing Acc1.

Loss Function	Balanced			Long-tailed			Head			Middle			Tail		
	Loss	Acc1	F1	Loss	Acc1	F1	Loss	Acc1	F1	Loss	Acc1	F1	Loss	Acc1	F1
Softmax	1.1455	0.7978	0.7967	1.1415	0.8059	0.8208	1.1208	0.8069	0.8587	1.3060	0.7870	0.8368	0.8690	0.8684	0.8684
Focal Loss	0.6765	0.8014	0.8001	0.7063	0.8011	0.8175	0.7005	0.7998	0.8531	0.8028	0.7870	0.8293	0.4055	0.8947	0.8860
Weighted Softmax	1.1455	0.7978	0.7967	1.1415	0.8059	0.8208	1.1208	0.8069	0.8587	1.3060	0.7870	0.8368	0.8690	0.8684	0.8684
Class-balanced	1.1455	0.7978	0.7967	1.1415	0.8059	0.8208	1.1208	0.8069	0.8587	1.3060	0.7870	0.8368	0.8690	0.8684	0.8684
Balanced Softmax	1.1289	0.8034	0.8011	1.1848	0.8030	0.8145	1.1469	0.8069	0.8553	1.4407	0.7574	0.8123	0.8872	0.9211	0.9298
Equalization	0.9992	0.7994	0.7983	1.0385	0.8040	0.8192	1.0118	0.8057	0.8564	1.2539	0.7692	0.8213	0.6035	0.9211	0.9211
LDAM	13.8126	0.7828	0.7817	13.5566	0.7821	0.7955	13.6884	0.7808	0.8325	14.7496	0.7574	0.8016	5.3231	0.9211	0.9035

Table A.2: Evaluation results for MobileNetV2 trained on the custom balanced dataset, showing Loss, Acc1, and F1 scores for each dataset split.

Loss Function	Balanced	Long-tailed	Head	Middle	Tail
Softmax	0.5282	0.7735	0.8341	0.5917	0.2368
Focal loss	0.5200	0.7745	0.8389	0.5917	0.1579
Weighted Softmax loss	0.5016	0.7231	0.7808	0.5503	0.2105
Class-balanced loss	0.1936	0.0913	0.0521	0.2485	0.2632
Balanced Softmax loss	0.5796	0.7650	0.8069	0.6331	0.4211
Equalization loss	0.5310	0.7650	0.8235	0.5917	0.2368
LDAM loss	0.4264	0.5899	0.6137	0.5444	0.2632

Table A.3: Evaluation results for MobileNetV2 trained on the long-tailed dataset showing Acc1.

Loss Function	Balanced			Long-tailed			Head			Middle			Tail		
	Loss	Acc1	F1	Loss	Acc1	F1	Loss	Acc1	F1	Loss	Acc1	F1	Loss	Acc1	F1
Softmax	3.2503	0.5282	0.4884	1.2212	0.7735	0.7578	0.8136	0.8341	0.8492	2.4604	0.5917	0.6444	4.7629	0.2368	0.2544
Focal Loss	2.3526	0.5200	0.4818	0.8022	0.7745	0.7602	0.5177	0.8389	0.8528	1.5864	0.5917	0.6625	3.6343	0.1579	0.1667
Weighted Softmax	3.1412	0.5016	0.4690	1.2817	0.7231	0.7104	0.8786	0.7808	0.8015	2.3365	0.5503	0.6213	5.1836	0.2105	0.1912
Class-balanced	4.3308	0.1936	0.1751	4.2181	0.0913	0.0854	4.4197	0.0521	0.0795	2.8788	0.2485	0.2767	6.3575	0.2632	0.2368
Balanced Softmax	3.1185	0.5796	0.5572	1.1630	0.7650	0.7685	0.7989	0.8069	0.8422	2.1612	0.6331	0.6872	4.8108	0.4211	0.4123
Equalization	3.0593	0.5310	0.4911	1.1563	0.7650	0.7499	0.7241	0.8235	0.8398	2.4487	0.5917	0.6524	4.9284	0.2368	0.2544
LDAM	21.4896	0.4264	0.3980	7.9893	0.5899	0.5909	5.6756	0.6137	0.6581	10.3379	0.5444	0.6121	49.1197	0.2632	0.2895

Table A.4: Evaluation results for MobileNetV2 trained on the long-tailed dataset, showing Loss, Acc1, and F1 scores for each dataset split.

A.2 ResNet50V2

ResNet50V2 trained on custom balanced dataset and imbalanced dataset on different loss functions.

Table A.5 show the top 1 accuracies for ResNet50V2 on various loss functions. Table A.6 show the loss, top 1 accuracy, and F1 score.

Table A.7 show the top 1 accuracies for ResNet50V2 on various loss functions. Table A.8 show the loss, top 1 accuracy, and F1 score.

Loss Function	Balanced	Long-tailed	Head	Middle	Tail
Softmax loss	0.8324	0.8421	0.8448	0.8047	0.9474
Focal loss	0.8310	0.8344	0.8341	0.8166	0.9211
Weighted Softmax loss	0.8324	0.8421	0.8448	0.8047	0.9474
Class-balanced loss	0.8324	0.8421	0.8448	0.8047	0.9474
Balanced Softmax loss	0.8310	0.8430	0.8460	0.8107	0.9211
Equalization loss	0.8292	0.8373	0.8412	0.7929	0.9474
LDAM loss	0.7990	0.7983	0.8069	0.7337	0.8947

Table A.5: Evaluation results for ResNet50V2 trained on the custom balanced dataset, showing Acc1.

Loss Function	Balanced			Long-tailed			Head			Middle			Tail		
	Loss	Acc1	F1	Loss	Acc1	F1	Loss	Acc1	F1	Loss	Acc1	F1	Loss	Acc1	F1
Softmax	0.9823	0.8324	0.8310	0.9874	0.8421	0.8520	0.9917	0.8448	0.8860	1.0934	0.8047	0.8467	0.4205	0.9474	0.9386
Focal Loss	0.5627	0.8310	0.8300	0.5578	0.8344	0.8474	0.5555	0.8341	0.8788	0.6294	0.8166	0.8607	0.2920	0.9211	0.9123
Weighted Softmax	0.9823	0.8324	0.8310	0.9874	0.8421	0.8520	0.9917	0.8448	0.8860	1.0934	0.8047	0.8467	0.4205	0.9474	0.9386
Class-balanced	0.9823	0.8324	0.8310	0.9874	0.8421	0.8520	0.9917	0.8448	0.8860	1.0934	0.8047	0.8467	0.4205	0.9474	0.9386
Balanced Softmax	1.0198	0.8310	0.8301	0.9689	0.8430	0.8549	0.9601	0.8460	0.8893	1.1309	0.8107	0.8539	0.4440	0.9211	0.9123
Equalization	0.8795	0.8292	0.8279	0.9079	0.8373	0.8495	0.8888	0.8412	0.8877	1.1374	0.7929	0.8453	0.2495	0.9474	0.9386
LDAM	9.8339	0.7990	0.7979	10.1092	0.7983	0.8119	9.8723	0.8069	0.8596	12.5229	0.7337	0.7823	4.6362	0.8947	0.8772

Table A.6: Evaluation results for ResNet50V2 trained on the custom balanced dataset, showing Loss, Acc1, and F1 scores for each dataset split.

Loss Function	Balanced	Long-tailed	Head	Middle	Tail
Softmax loss	0.5522	0.7954	0.8531	0.6391	0.2105
Focal loss	0.5456	0.7935	0.8483	0.6272	0.3158
Weighted Softmax loss	0.4976	0.7336	0.7915	0.5562	0.2368
Class-balanced loss	0.2052	0.1836	0.1445	0.3787	0.1842
Balanced Softmax loss	0.5908	0.7916	0.8270	0.6568	0.6053
Equalization loss	0.5452	0.7897	0.8389	0.6450	0.3421
LDAM loss	0.3742	0.5937	0.6469	0.4438	0.0789

Table A.7: Evaluation results for ResNet50V2 trained on the long-tailed dataset, showing Acc1.

Loss Function	Balanced			Long-tailed			Head			Middle			Tail		
	Loss	Acc1	F1	Loss	Acc1	F1	Loss	Acc1	F1	Loss	Acc1	F1	Loss	Acc1	F1
Softmax	3.0907	0.5522	0.5138	1.0524	0.7954	0.7798	0.6888	0.8531	0.8654	2.0330	0.6391	0.6996	4.7658	0.2105	0.2018
Focal Loss	2.0718	0.5456	0.5089	0.6284	0.7935	0.7789	0.3983	0.8483	0.8583	1.3258	0.6272	0.7054	2.6364	0.3158	0.3158
Weighted Softmax	3.7904	0.4976	0.4591	1.3481	0.7336	0.7198	0.8630	0.7915	0.8098	2.2625	0.5562	0.6209	6.9808	0.2368	0.2456
Class-balanced	4.5887	0.2052	0.1928	3.7422	0.1836	0.1932	3.7880	0.1445	0.2045	2.4884	0.3787	0.4138	8.3052	0.1842	0.1737
Balanced Softmax	3.1081	0.5908	0.5654	1.0452	0.7916	0.7895	0.6873	0.8270	0.8602	2.3422	0.6568	0.7135	3.2275	0.6053	0.5965
Equalization	3.0166	0.5452	0.5071	1.0315	0.7897	0.7756	0.7418	0.8389	0.8511	1.8754	0.6450	0.7061	3.6342	0.3421	0.3509
LDAM	22.7933	0.3742	0.3337	8.2056	0.5937	0.5784	5.3320	0.6469	0.6680	12.3074	0.4438	0.5450	53.4080	0.0789	0.0789

Table A.8: Evaluation results for ResNet50V2 trained on the long-tailed dataset, showing Loss, Acc1, and F1 scores for each dataset split.

A.3 ViT-B/16

ViT-B/16 trained on custom balanced dataset and imbalanced dataset on different loss functions.

Table A.9 show the top 1 accuracies for ViT-B/16 on various loss functions. Table A.10 show the loss, top 1 accuracy, and F1 score.

Table A.11 show the top 1 accuracies for ViT-B/16 on various loss functions. Table A.12 show the loss, top 1 accuracy, and F1 score.

Loss Function	Balanced	Long-tailed	Head	Middle	Tail
Softmax loss	0.5620	0.5671	0.5521	0.6036	0.7368
Focal loss	0.5516	0.5538	0.5438	0.5680	0.7105
Weighted Softmax loss	0.5620	0.5671	0.5521	0.6036	0.7368
Class-balanced loss	0.5620	0.5671	0.5521	0.6036	0.7368
Balanced Softmax loss	0.5628	0.5642	0.5640	0.5325	0.7105
Equalization loss	0.5634	0.5519	0.5462	0.5503	0.6842
LDAM loss	0.5906	0.6013	0.5924	0.6095	0.7632

Table A.9: Evaluation results for ViT-B/16 trained on the custom balanced dataset, showing Acc1.

Loss Function	Balanced			Long-tailed			Head			Middle			Tail		
	Loss	Acc1	F1	Loss	Acc1	F1	Loss	Acc1	F1	Loss	Acc1	F1	Loss	Acc1	F1
Softmax	4.6431	0.5620	0.5593	4.6089	0.5671	0.5951	4.6420	0.5521	0.6367	4.7263	0.6036	0.6648	3.3521	0.7368	0.7281
Focal Loss	2.3473	0.5516	0.5488	2.3562	0.5538	0.5869	2.4288	0.5438	0.6324	2.2330	0.5680	0.6355	1.2929	0.7105	0.6930
Weighted Softmax	4.6431	0.5620	0.5593	4.6089	0.5671	0.5951	4.6420	0.5521	0.6367	4.7263	0.6036	0.6648	3.3521	0.7368	0.7281
Class-balanced	4.6431	0.5620	0.5593	4.6089	0.5671	0.5951	4.6420	0.5521	0.6367	4.7263	0.6036	0.6648	3.3521	0.7368	0.7281
Balanced Softmax	4.7131	0.5628	0.5592	4.6809	0.5642	0.5929	4.7739	0.5640	0.6471	4.8161	0.5325	0.5998	2.0138	0.7105	0.7105
Equalization	4.2603	0.5634	0.5614	4.4906	0.5519	0.5884	4.6109	0.5462	0.6410	4.3952	0.5503	0.6014	2.0079	0.6842	0.6754
LDAM	48.2745	0.5906	0.5926	47.4149	0.6013	0.6348	49.6692	0.5924	0.6790	42.0117	0.6095	0.6780	21.3751	0.7632	0.7281

Table A.10: Evaluation results for ViT-B/16 trained on the custom balanced dataset, showing Loss, Acc1, and F1 scores for each dataset split.

Loss Function	Balanced	Long-tailed	Head	Middle	Tail
Softmax loss	0.2254	0.4367	0.5071	0.1775	0.0263
Focal loss	0.2210	0.4206	0.4834	0.1953	0.0263
Weighted Softmax loss	0.1284	0.1760	0.1919	0.1302	0.0263
Class-balanced loss	0.0558	0.0076	0.0000	0.0237	0.1053
Balanced Softmax loss	0.2460	0.4244	0.4822	0.2130	0.0789
Equalization loss	0.2168	0.4215	0.4893	0.1716	0.0263
LDAM loss	0.5906	0.6013	0.5924	0.6095	0.7632

Table A.11: Evaluation results for ViT-B/16 trained on the long-tailed dataset, showing Acc1.

Loss Function	Balanced			Long-tailed			Head			Middle			Tail		
	Loss	Acc1	F1	Loss	Acc1	F1	Loss	Acc1	F1	Loss	Acc1	F1	Loss	Acc1	F1
Softmax	13.5272	0.2254	0.1871	6.7999	0.4367	0.4216	5.3024	0.5071	0.5248	11.3663	0.1775	0.2303	19.7511	0.0263	0.0263
Focal Loss	7.5701	0.2210	0.1850	3.6474	0.4206	0.4016	2.8064	0.4834	0.4914	6.1246	0.1953	0.2666	11.3091	0.0263	0.0263
Weighted Softmax	6.5391	0.1284	0.1144	3.9782	0.1760	0.1902	3.4559	0.1919	0.2357	4.7288	0.1302	0.1541	11.0975	0.0263	0.0351
Class-balanced	4.9938	0.0558	0.0368	5.8487	0.0076	0.0028	6.2065	0.0000	0.0000	4.7503	0.0237	0.0292	4.0694	0.1053	0.0746
Balanced Softmax	13.3583	0.2460	0.2123	6.7016	0.4244	0.4175	5.2929	0.4822	0.5121	11.3472	0.2130	0.2710	17.3287	0.0789	0.0877
Equalization	13.4511	0.2168	0.1786	6.7202	0.4215	0.4062	5.2340	0.4893	0.5051	11.6755	0.1716	0.2353	17.4650	0.0263	0.0263
LDAM	48.2745	0.5906	0.5926	47.4149	0.6013	0.6348	49.6692	0.5924	0.6790	42.0117	0.6095	0.6780	21.3751	0.7632	0.7281

Table A.12: Evaluation results for ViT-B/16 trained on the long-tailed dataset, showing Loss, Acc1, and F1 scores for each dataset split.

A.4 ConvNeXt Base

ConvNeXt Base trained on custom balanced dataset and imbalanced dataset on different loss functions.

Table A.13 show the top 1 accuracies for ConvNeXt Base on various loss functions. Table A.14 show the loss, top 1 accuracy, and F1 score.

Table A.15 show the top 1 accuracies for ConvNeXt Base on various loss functions. Table A.16 show the loss, top 1 accuracy, and F1 score.

Loss Function	Balanced	Long-tailed	Head	Middle	Tail
Softmax loss	0.8332	0.8535	0.8566	0.8166	0.9474
Focal loss	0.8314	0.8487	0.8507	0.8284	0.8947
Weighted Softmax loss	0.8332	0.8535	0.8566	0.8166	0.9474
Class-balanced loss	0.8332	0.8535	0.8566	0.8166	0.9474
Balanced Softmax loss	0.8364	0.8344	0.8365	0.7988	0.9474
Equalization loss	0.8318	0.8468	0.8448	0.8343	0.9474
LDAM loss	0.8316	0.8373	0.8412	0.8047	0.8947

Table A.13: Evaluation results for ConvNeXt Base trained on the custom balanced dataset, showing Acc1.

Loss Function	Balanced			Long-tailed			Head			Middle			Tail		
	Loss	Acc1	F1	Loss	Acc1	F1	Loss	Acc1	F1	Loss	Acc1	F1	Loss	Acc1	F1
Softmax	0.9904	0.8332	0.8323	0.9594	0.8535	0.8661	0.9571	0.8566	0.9010	1.1028	0.8166	0.8603	0.3731	0.9474	0.9386
Focal Loss	0.5686	0.8314	0.8301	0.5597	0.8487	0.8608	0.5640	0.8507	0.8975	0.6046	0.8284	0.8730	0.2629	0.8947	0.8947
Weighted Softmax	0.9904	0.8332	0.8323	0.9594	0.8535	0.8661	0.9571	0.8566	0.9010	1.1028	0.8166	0.8603	0.3731	0.9474	0.9386
Class-balanced	0.9904	0.8332	0.8323	0.9594	0.8535	0.8661	0.9571	0.8566	0.9010	1.1028	0.8166	0.8603	0.3731	0.9474	0.9386
Balanced Softmax	1.0008	0.8364	0.8350	0.9829	0.8344	0.8478	0.9720	0.8365	0.8853	1.1509	0.7988	0.8418	0.4780	0.9474	0.9386
Equalization	0.9124	0.8318	0.8302	0.9030	0.8468	0.8594	0.8550	0.8448	0.8899	1.2187	0.8343	0.8779	0.4981	0.9474	0.9474
LDAM	12.2036	0.8316	0.8308	11.0787	0.8373	0.8485	10.9948	0.8412	0.8882	12.5744	0.8047	0.8592	6.2892	0.8947	0.8947

Table A.14: Evaluation results for ConvNeXt Base trained on the custom balanced dataset, showing Loss, Acc1, and F1 scores for each dataset split.

Text.

Loss Function	Balanced	Long-tailed	Head	Middle	Tail
Softmax loss	0.5972	0.8316	0.8898	0.6568	0.3158
Focal loss	0.5938	0.8145	0.8685	0.6568	0.3158
Weighted Softmax loss	0.4090	0.6356	0.6848	0.4911	0.1842
Class-balanced loss	0.0142	0.0019	0.0000	0.0000	0.0526
Balanced Softmax loss	0.6460	0.8230	0.8685	0.6509	0.5789
Equalization loss	0.5956	0.8278	0.8768	0.6923	0.3421
LDAM loss	0.3770	0.5956	0.6445	0.4260	0.2632

Table A.15: Evaluation results for ConvNeXt Basetrained on the long-tailed dataset, showing Acc1.

Loss Function	Balanced			Long-tailed			Head			Middle			Tail		
	Loss	Acc1	F1	Loss	Acc1	F1	Loss	Acc1	F1	Loss	Acc1	F1	Loss	Acc1	F1
Softmax	2.7006	0.5972	0.5645	0.9552	0.8316	0.8202	0.5867	0.8898	0.9013	2.1181	0.6568	0.7177	3.9670	0.3158	0.3158
Focal Loss	1.8210	0.5938	0.5615	0.6024	0.8145	0.8002	0.3485	0.8685	0.8791	1.3247	0.6568	0.7197	3.0291	0.3158	0.3070
Weighted Softmax	4.5284	0.4090	0.3763	2.0092	0.6356	0.6266	1.5444	0.6848	0.7054	3.1309	0.4911	0.5827	6.0533	0.1842	0.1930
Class-balanced	5.0105	0.0142	0.0016	6.1643	0.0019	0.0000	6.5523	0.0000	0.0000	5.0450	0.0000	0.0000	3.4200	0.0526	0.0164
Balanced Softmax	2.6574	0.6460	0.6273	0.9120	0.8230	0.8237	0.5457	0.8685	0.8952	2.1945	0.6509	0.7250	3.3453	0.5789	0.5965
Equalization	2.5527	0.5956	0.5586	0.9349	0.8278	0.8139	0.6192	0.8768	0.8907	1.9792	0.6923	0.7375	3.2293	0.3421	0.3404
LDAM	39.0426	0.3770	0.3448	12.8480	0.5956	0.5812	8.3491	0.6445	0.6597	25.5813	0.4260	0.5105	72.0081	0.2632	0.2719

Table A.16: Evaluation results for ConvNeXt Base trained on the long-tailed dataset, showing Loss, Acc1, and F1 scores for each dataset split.