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

Contents

Chapter 1

Introduction

TODO: Emphasize head and tail classes.
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 TODO: reference here. The trained models have shown image classification with accuracies of over 80 % TODO: 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 TODO: reference here, while most real-world datasets are skewed in samples per class TODO: 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

TODO: 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 ImageNet, 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

TODO: finish this section. 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 re-weighting as a solution to handle imbalanced datasets. The loss re-weighting methods include cross-entropy loss, focal loss, weighted cross-entropy loss, class-balanced loss, balanced softmax loss, equalization loss, and LDAM loss. The experiments are conducted on the CIFAR100 dataset, with both a balanced and synthetically generated long-tailed version. 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. TODO: not yet module improvement.

1.2 Reading Guide

TODO: Mention what each chapter will cover and how they relate to each other.

1.3 Related Work

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

1.3.1 Data Re-Balancing

1.3.2 Re-Weighting

1.3.3 Margin Loss

1.3.4 Meta-Learning

1.3.5 Decoupled Training

1.3.6 Transfer Learning

1.3.7 Ensemble Learning

1.3.8 Few Shot Learning

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 *Vision 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.
TODO: 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.

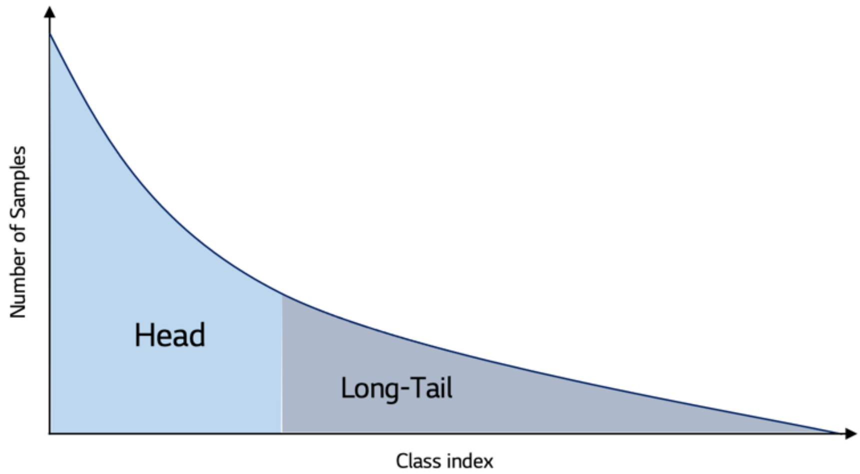

Figure 2.1: Illustration of a long-tailed distribution. Figure from [10].

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 Architecures

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 Vision Transformers (ViTs) are chosen as the primary architectures used in this thesis due to their performance on image classification tasks. 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 ViT architecture.

TODO: A brief summary of the relevance of the chosen architectures so that the reader knows why they are included.

2.2.1 Introduction 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, 18].

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.

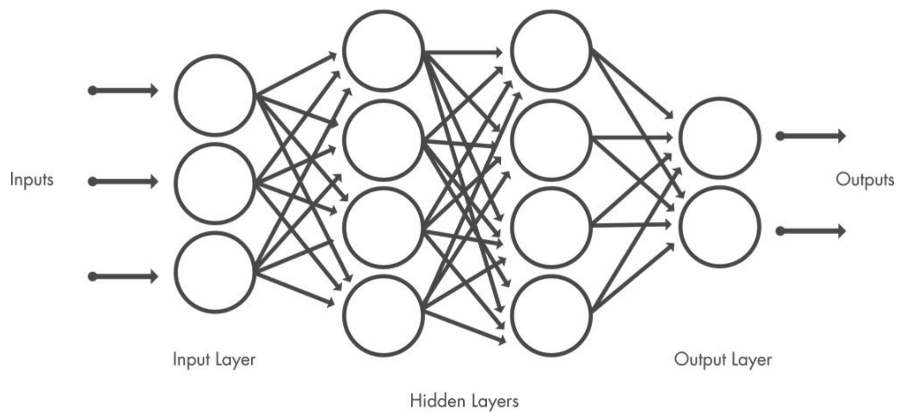

Figure 2.2: Layers of a neural network. Figure from [19]. TODO: Make this figure.

However, this simple neural network becomes insufficient for more complex problems, such as image classification, as it requires an increasing number of parameters. For instance, a 224×224 RGB image flattened is very large, making MLPs parameter-heavy and inefficient. The limitations of MLPs were addressed by Convolutional Neural Networks (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

TODO: finish this section.
Convolutional Neural Networks (CNNs) [20] were introduced to address the limitations of MLPs for image-related tasks. Unlike MLPs, which treat input features as independent, CNNs are designed to recognize patterns in images,by applying local filters through convolutional layers, and thereby preserving the two dimensional input of an image, preserving the idea that nearby pixels are related [21, 22, 17].
CNNs gained popularity after the introduction of LeNet-5 by LeCun et al. in 1998 [21] that demonstrated the potential of CNNs by recognizing handwritten digits. Later, AlexNet [22] achieved a breakthrough by winning the ImageNet Challenge in 2012, demonstrating the potential of CNNs to handle large-scale image recognition tasks by deepening the architechtures and utilizing multiple GPUs for training. Subsequently, the evolution of CNNs has progressed through architectures such as VGGNet [23], GoogLeNet [24], and ResNet [14], which have set the stage for the advancements seen in modern CNNs.
CNNs consist of several core components, as illustrated in Figure 2.3. Convolutional layers extract local features by applying filters to small regions of an image, while pooling layers reduce the spatial dimensions of feature maps, providing invariance to small translations. Activation functions introduce nonlinearity, allowing the network to capture complex patterns. At the final stage, fully connected or global pooling layers aggregate the extracted features into predictions, enabling tasks such as classification or segmentation.

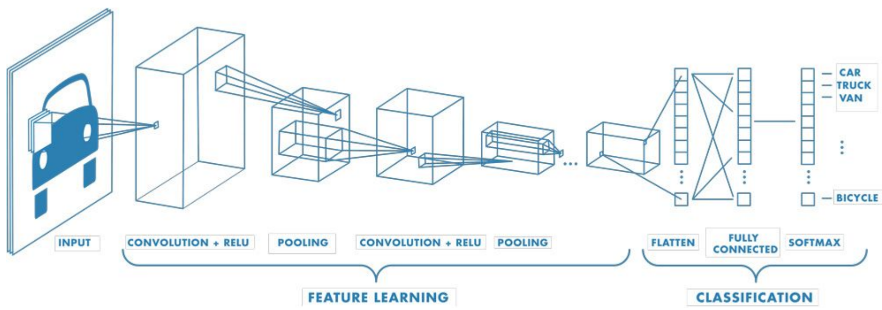

Figure 2.3: Illustration of a convolutional neural network. Figure from [19]. TODO: Make this figure.

TODO: Tie to the models used for experiments in this thesis.

ResNet50V2 Architecture

TODO: write this section. 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: identity mapping, and reordering of batch normalization and relu activation [25].
ResNet50V2 is the 50-layer version of the ResNet architecture.

Improvement from predecessors: addressing vanishing gradients.

MobileNetV2 Architecture

MobileNetV2 introduced by Sandler et al. [13] is a lightweight CNN model designed primarily to balance model accuracy and computational efficiency, making it suitable for mobile or embedded devices. Building upon the original concepts of MobileNetV1 [26], MobileNetV2 preserves the use of depthwise separable convolutions, a method for reducing the parameters and floating-point operations, while introducing a novel element known as the inverted residual structure.

Inverted Residual Blocks While traditional residual connections, introduced in Residual Networks (ResNets) [14] TODO: as discussed in the previous section, allow for identity mapping and improved gradient flow, MobileNetV2 employs an inverted residual structure [13]. Instead of mapping from a high-dimensional representation down to a lower-dimensional bottleneck, then reconstructing features at the output, inverted residual blocks begin with a low-dimensional input and expand it to a higher-dimensional space before applying a depthwise convolution. After spatial filtering, the representation is projected back down to a low-dimensional space. This approach, illustrated in Figure 2.4, helps preserve crucial information and maintain a rich feature space without substantially increasing computational cost. The use of a linear bottleneck (i.e., no nonlinear activation in the low-dimensional projection) also helps prevent the destruction of useful information, further improving efficiency and accuracy.

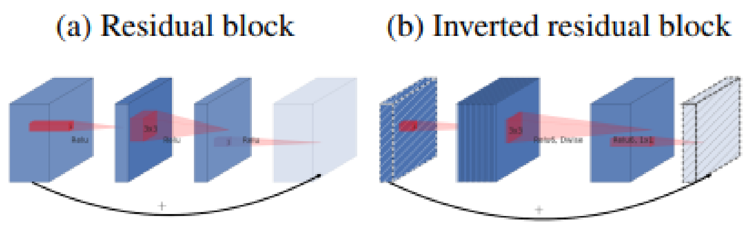

Figure 3: The difference between residual block [8, 30] and inverted residual. Diagonally hatched layers do not use non-linearities. We use thickness of each block to indicate its relative number of channels. Note how classical residuals connects the layers with high number of channels, whereas the inverted residuals connect the bottlenecks. Best viewed in color.

Figure 2.4: Residual and Inverted Residual Blocks. Figure from [13]. TODO: Make this figure.

Depthwise Separable Convolution Following MobileNetV1, MobileNetV2 relies on depthwise separable convolutions to factorize the convolution operation into two simpler operations [26]: depthwise convolution and pointwise convolution as shown in figure 2.5. The depthwise convolution applies a single filter to each input channel, and the pointwise convolution (a 1×1 convolution) then recombines the channels to produce the desired output. In comparison, a standard convolution both filtes and combines inputs into a new set of outputs. This approach reduces the parameter count and computational load, making the model suitable for devices with limited resources [26, 13].

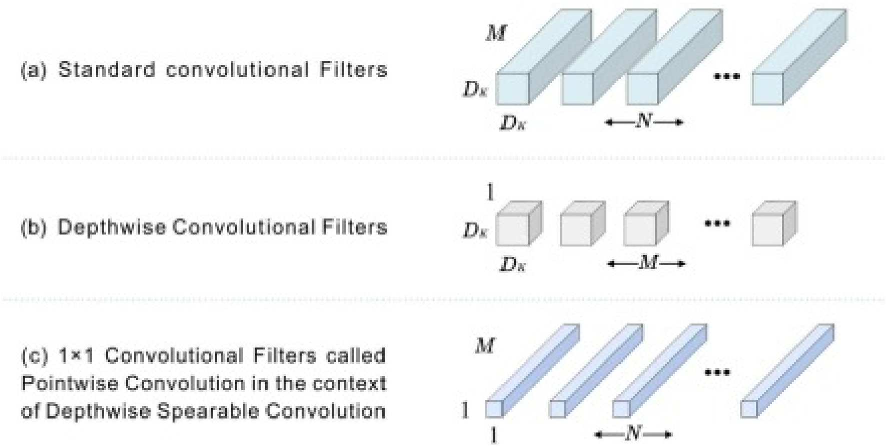

Figure 2.5: Depthwise Separable Convolution. Figure from `https://www.sciencedirect.com/topics/computer-science/depthwise-separable-convolution`. TODO: Make this figure.

Relevance In this thesis, MobileNetV2 represents an example of a modern, efficient CNN architecture. Its lightweight design makes it particularly attractive for applications where computational resources are limited. By including MobileNetV2 among the evaluated architectures, the performance is compared across varying complexity levels, providing insights into how efficiency-oriented designs fare against more complex models. This comparison is especially relevant if the application domain involves real-time processing or deployment on mobile or embedded devices.

ConvNeXt Base Architecture

TODO: Write this section. 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.

Improvement from predecessors: integrating insights from ViTs.

"The "Roaring 20s" of visual recognition began with the introduction of Vision Transformers (ViTs), which quickly superseded ConvNets as the state-of-the-art image classification model. A vanilla ViT, on the other hand, faces difficulties when applied to general computer vision tasks such as object detection and semantic segmentation. It is the hierarchical Transformers (e.g., Swin Transformers) that reintroduced several ConvNet priors, making Transformers practically viable as a generic vision backbone and demonstrating remarkable performance on a wide variety of vision tasks. However, the effectiveness of such hybrid approaches is still largely credited to the intrinsic superiority of Transformers, rather than the inherent inductive biases of convolutions. In this work, we reexamine the design spaces and test the limits of what a pure ConvNet can achieve. We gradually "modernize" a standard ResNet toward the design of a vision Transformer, and discover several key components that contribute to the performance difference along the way. The outcome of this exploration is a family of pure ConvNet models dubbed ConvNeXt. Constructed entirely from standard ConvNet modules, ConvNeXts compete favorably with Transformers in terms of accuracy and scalability, achieving 87.8 % ImageNet top-1 accuracy and outperforming Swin Transformers on COCO detection and ADE20K segmentation, while maintaining the simplicity and efficiency of standard ConvNets." `https://paperswithcode.com/paper/a-convnet-for-the-2020s`

2.2.3 Vision Transformers

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

ViT-B/16 Architecture

TODO: write this section. ViT-B/16 is a Vision 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. `https://sh-tsang.medium.com/review-vision-transformer-vit-406568`

The Vision Transformer, or ViT, is a model for image classification that employs a Transformer-like architecture over patches of the image. An image is split into fixed-size patches, each of them are then linearly embedded, position embeddings are added, and the resulting sequence of vectors is fed to a standard Transformer encoder. In order to perform classification, the standard approach of adding an extra learnable "classification token" to the sequence is used. `https://paperswithcode.com/method/vision-transformer`

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 2.6. 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.

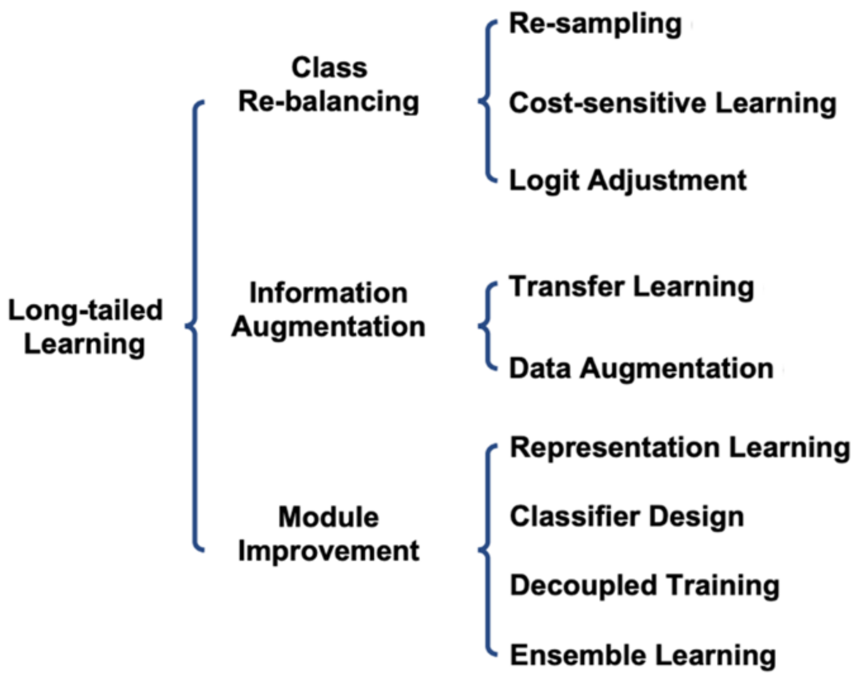

Figure 2.6: Long-tailed categories as described by *Zhang et al.* [1].

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.

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 [27] [28].
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 [28] [29]. 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) [29], 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) [30], ...

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

Equalization Loss *Equalization loss*, introduced by Tan et al. (2020) [31], ...

2.3.2 Information Augmentation

Data augmentation techniques tailored for long-tailed datasets.

Transfer Learning

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 [32] 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.

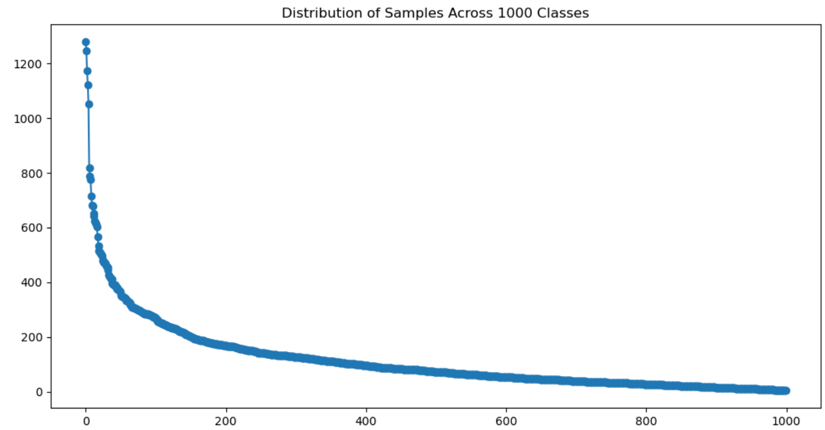

Figure 3.1: The class distribution of the training images for the ImageNet-LT dataset shows a long-tailed distribution.

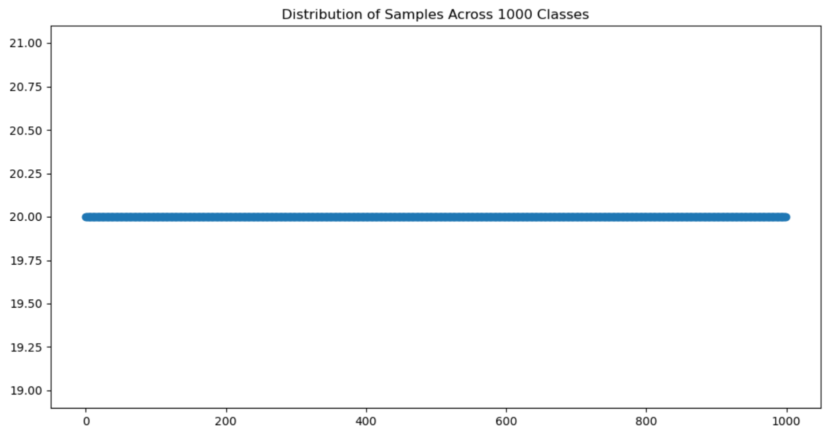

Figure 3.2: The class distribution of the validation images for the ImageNet-LT dataset shows that there are 20 samples of each class.

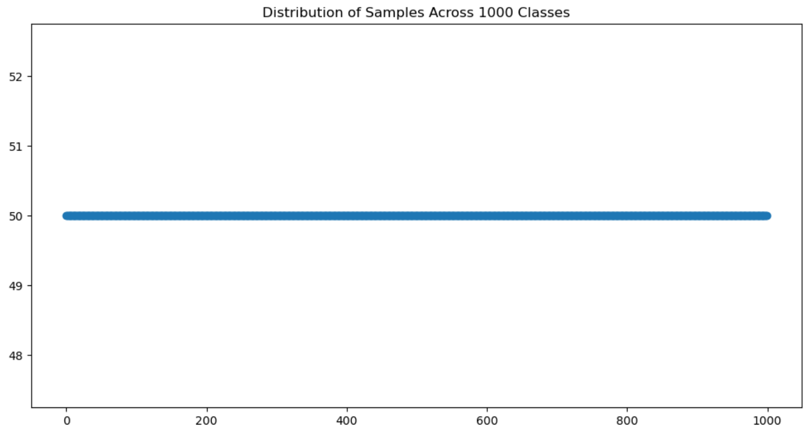

Figure 3.3: The class distribution of the test images for the ImageNet-LT dataset shows that there are 50 samples of each class.

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.

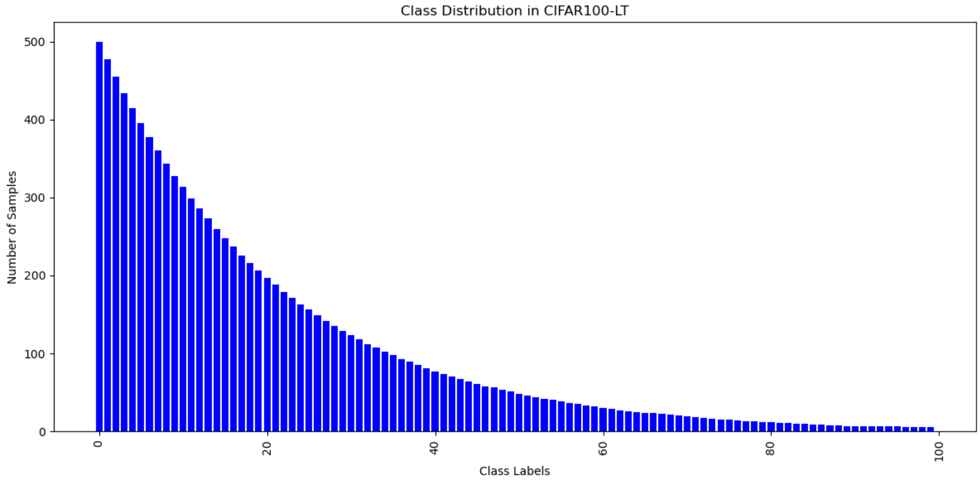

Figure 3.4: The class distribution of CIFAR100-LT with imbalance ratio 100 generated by the imbalance_cifar.py in the LDAM-DRW GitHub repository [9].

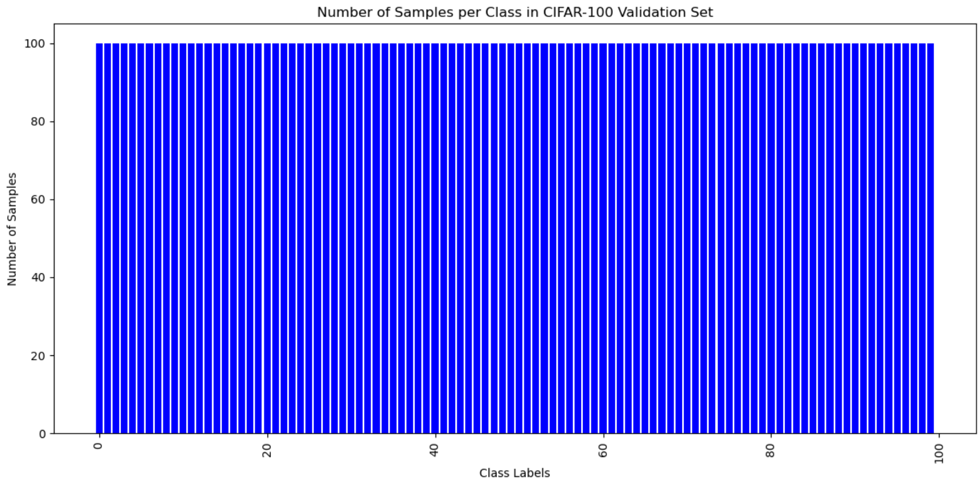

Figure 3.5: The class distribution of the CIFAR100 validation set from torchvision [reference here].

TODO: Include plots of head, middle and tail test splits.

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, potentially 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.

3.4.1 Model Comparison

Metrics for comparison of overall model performance. Example: harmonic mean, geometric mean, min score. Mention why these metrics are suitable for balancing performance across head, middle, and tail classes.

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 into training and test sets. The test set derived from the split consists of 5,000 samples, with 50 samples per class. The validation set, unaltered from the original CIFAR-100 test set, contains 10,000 samples with an equal distribution of 100 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 sample specifics are presented in table 4.1.

4.2 Data Preprocessing

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

Table 4.1: 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: investigate	Variable
Middle Test Set	TODO: investigate	Variable
Tail Test Set	TODO: investigate	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

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. 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.

MobileNetV2 The MobileNetV2 architecture is pretrained on ImageNet-1K dataset, and the classifciation layer was replaced with a 100-class fully connected layer. TODO: reference

ResNet50V2 The ResNet50V2 architecture is pretrained on ImageNet-1K dataset, and the fc-layer is replaced with a 100-class fully connected layer. TODO:

reference

ViT-B/16 The ViT-B/16 architecture is pretrained on the ImageNet-21K dataset and fine-tuned on the ImageNet-1K dataset [33]. The head is replaced with a 100-clas fully connected layer.

ConvNeXt Base The ConvNeXt Base architecture is pretrained on the ImageNet-1K dataset [34], and the final layer is replaced with a 100-class fully connected layer.

The specifications of the model architectures can be seen in table 4.4 TODO: make this table prettier.

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

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**: Initial value of `0.001`.
- **Learning Rate Scheduler**: StepLR with a step size of `30` epochs and a decay factor of `0.1`.
- **Batch Size**: `128`.
- **Number of Epochs**: `90`.
- **Weight Decay**: Default value of `1e-4`.
- **Class Weights**: Dynamically computed based on the training dataset and passed to the loss function. TODO: Reference to Methodology section.
- **Number of GPUs**: 4. See section 4.6 for specifications.
- **Device Setup**: Utilized `torch.nn.DataParallel` for multi-GPU training.

- **Checkpoint Criteria**: The best model is saved based on the highest Top-1 validation accuracy.

4.5 Evaluation Metrics

The primary metric used to evaluate model performance during validation and testing is the top-1 accuracy. Alongside, the F1 score was calculated (macro F1 for balanced datasets, and weighted F1 for imbalanced dataset) but never used for evaluation. The F1 scores for all experiments can be seen in appendix A.

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 reusing the configuration files.

- **Saved Artifacts**:
  - Datasets were saved, making them accessible for evaluation or reuse in future experiments.
  - Model checkpoints were saved after achieving the best validation accuracy.

4.8 Implementation Faults

TODO: Revisit this section later on.

- The wrong version of the ViT-B/16 architecture might have implemented. The version implemented is the vit-base-patch16-224 [33] via timm which was pretrained on the ImageNet-21K and later fine-tuned on ImageNet-1K, whereas the other models are pretrained solely on the ImageNet-1K. The version on ViT-B/16 that is pretrained on the ImageNet-1K can be implemented as the torchvision version vit_b_16 [35].

- Class-Balanced Loss.

Chapter 5

Results and Discussion

This chapter presents the experimental results, comparing model performances across head, middle, and tail classes, and discussing the impact of different model architechtures combined with diffrent loss functions. First, the main findings are presented, followed by a detailed analysis of the results across experiments, and lastly a discussion of the results.

5.1 Main Findings

Across all models, Balanced Softmax Loss demonstrated the highest performance on tail classes for models trained on long-tailed datasets while maintaining consistent performance on head, middle, and overall long-tailed test sets. The highest top-1 accuracy for tail classes was achieved by the ResNet50V2 architecture (Acc1: 0.6053), closely followed by the ConvNeXt Base architecture (Acc1: 0.5789). However, this improved tail-class performance comes at the cost of head-class accuracy, where ConvNeXt Base outperforms ResNet50V2 with a top-1 accuracy of 0.8685 compared to 0.8270. Overall, ConvNeXt Base demonstrates better performance across all classes (Acc1: 0.8230) compared to ResNet50V2 (Acc1: 0.7916). See Tables 5.4 and 5.8 for reference. These results, however, require further statistical analysis to establish their significance.
Class-Balanced Loss consistently underperformed, warranting further investigation into its implementation. Similarly, the ViT-B/16 architecture demonstrated the lowest overall accuracy when trained on both balanced and long-tailed datasets (Acc1: 59.06 %, see Table 5.5), despite having the highest reported benchmark performance (Acc1: 93.95 %) among all model architectures investigated in this thesis [36]. This discrepancy suggests potential limitations in its design or configuration.

5.2 Overall Results

This section presents the overall results of all experiments conducted in this thesis, commenting on the best and worst performance of loss designs on a given model, and not directly comparing loss designs or models. This section is meant as an overview of all findings.

5.2.1 MobileNetV2

Results from Balanced Training Dataset

Table 5.1: Top-1 accuracy results for MobileNetV2 on the balanced dataset across all 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	**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**

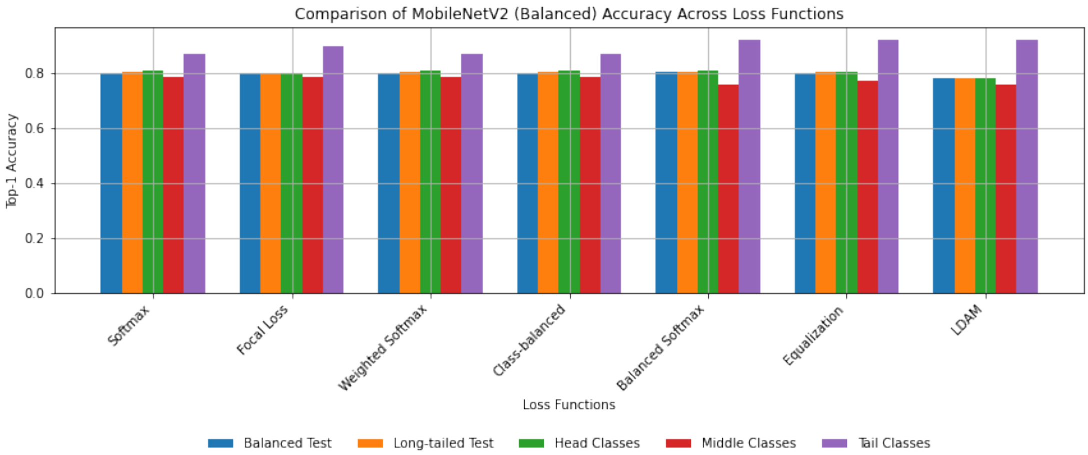

Figure 5.1: Top-1 accuracy comparison of MobileNetV2 trained on the balanced CIFAR-100 across loss functions on different test datasets. The bars represent performance on balanced test data, long-tailed test data, and head, middle, and tail classes.

TODO: Comment on 5.1.
From Table 5.1, the overall best performance on MobileNetV2 trained with a balanced CIFAR100 training dataset is achieved by Balanced Softmax Loss, which has the highest accuracy on the balanced test dataset (Acc1: 0.8034), as well as on the head (Acc1: 0.8069) and tail (Acc1: 0.9211) classes, with only slightly

worse performance on the middle classes in comparison. Among all loss functions, LDAM Loss shows the lowest overall performance on the balanced test set (Acc1: 0.7828) and the long-tailed test set (Acc1: 0.7821), except for its strong performance on tail classes (Acc1: 0.9211).
Softmax Loss, Weighted Softmax Loss, and Class-Balanced Loss yield the same accuracies across all test datasets, likely due to their similarities in loss design TODO: Refer to background section.
Balanced Softmax Loss, Equalization Loss, and LDAM Loss exhibit the highest accuracy on tail classes (Acc1: 0.9211). Despite their differing loss designs, this convergence in accuracy suggests that the dataset's tail-class performance may have reached a plateau, possibly due to the inherent characteristics of the tail classes, i.e. either noise or the limited number of samples available per class in the tail TODO: Refer to background section.
A similar trend is observed in the middle-class accuracy, where Softmax, Focal Loss, Weighted Softmax Loss, and Class-Balanced Loss all achieve identical results (Acc1: 0.7870). Similarly, for head classes, Softmax, Weighted Softmax Loss, Class-Balanced Loss, and Balanced Softmax Loss perform equally well, achieving the highest accuracy (Acc1: 0.8069).

Results from Long-Tailed Training Dataset

Table 5.2 shows the top 1 accuracies for MobileNetV2 on all loss functions.

Table 5.2: Top-1 accuracy results for MobileNetV2 on the long-tailed dataset across all 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

TODO: Comment on figure 5.2.
From table 5.2, the best overall performance on MobileNetV2 trained with a long-tailed CIFAR100 training dataset is achieved by Balanced Softmax Loss, with the highest accuracy on the balanced test set (Acc1: 0.5796), middle classes (Acc1: 0.6331), and tail classes (Acc1: 0.4211), and with competing accuracies on the long-tailed test dataset (Acc1: 0.7650) and head classes (Acc1: 0.8069), with the highest performance of Focal Loss with top-1 accuracies of 0.7745 and 0.8389, respectively.
The Class-Balanced loss exhibits the least satisfactory accuracies across all test dataset (Balanced: 0.1936, Long-Tailed: 0.0913, Head: 0.0521, Middle: 0.2485),

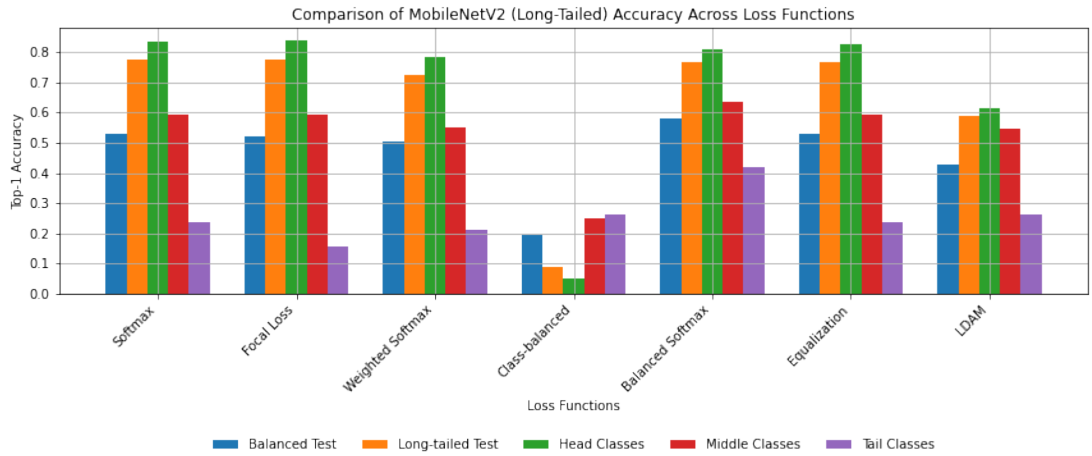

Figure 5.2: Top-1 accuracy comparison of MobileNetV2 trained on the long-tailed CIFAR-100 across loss functions on different test datasets. The bars represent performance on balanced test data, long-tailed test data, and head, middle, and tail classes.

with the only exception at tail classes (Acc1: 0.2632), where the results are comparable to those of other loss designs. This is a contrast to the performance when trained with a balanced CIFAR100 dataset, where the loss design performed within range of the other loss designs, possibly indicating a fault in implementation TODO: investigate implementation.

Unlike the performance of loss functions trained on the balanced CIFAR100, there are not as many incidents of accuracies of the same value, except for the performance of Softmax Loss, Focal Loss, and Equalization loss on middle classes (Acc1: 0.5917). However, this value is not the highest, as the Balanced Softmax Loss achieves a top-1 accuracy of 0.6331. TODO: explain this.

Comparison to Benchmark

TODO: No benchmark for MobileNetV2 trained on CIFAR-100. The paper *BiTAT: Neural Network Binarization with Task-dependent Aggregated Transformation* by Park et al. tested a MobileNetV2 architecture on CIFAR100 with the highest top 1 accuracy of 73.20 % . Link to paper: `https://arxiv.org/pdf/2207.01394`

The paper *E^2-Train: Training State-of-the-art CNNs with Over 80% Energy Savings* by Wang et al. reported a top 1 accuracy of 71.91 % with MobileNetV2 evaluated on CIFAR100. Link to paper: `https://arxiv.org/pdf/1910.13349`

Mine is 80.34 %. See table 5.1.

5.2.2 ResNet50V2

Results from Balanced Training Dataset

Table 5.3 show the top 1 accuracies for ResNet50V2 on all loss functions.

Table 5.3: Evaluation results for ResNet50V2 trained on the custom balanced dataset, showing Acc1.

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

From table 5.3, there are three loss designs with the best perfomance on the balanced test dataset, namely Softmax Loss, Weighted Softmax Loss, and Class-Balanced Loss (Acc1: 0.8324). Furthermore, they all yeild the same results across all test dataset due to their cross-entropy architecture for balanced training data. Likewise they yield the best perfomance on the tail classes along with equalization loss (Acc1: 0.9474) TODO: explain.
The perfomance of the Balanced Softmax Loss shows excellence on the long-tailed dataset (Acc1: 0.8430) as well as the head classes (Acc1: 0.8460) with competing results on both middle (Acc1: 0.8107) and tail (Acc1: 0.9211) classes.
The worst performance is that of the LDAM loss across all test datasets.

Results from Long-Tailed Training Dataset

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

Table 5.4: Evaluation results for ResNet50V2 trained on the long-tailed dataset, showing Acc1.

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

From table 5.4, the performance of the Balanced Softmax Loss present the best on the balanced dataset (Acc1: 0.5906), middle (Acc1: 0.6568), and tail classes (Acc1: 0.6053) with competing perfomances on the long-tailed datsaet (Acc1: 0.7916) and head classes (Acc1: 0.8270). The best performance on the long-tailed dataset and head classes was presented by the Softmax Loss (Acc1: 0.7954, 0.8531).
Two loss designs are competing for the worst performance with underwhelming results across all test datasets, namely the Class-Balanced loss and the LDAM loss. The LDAM loss achieved an accuracy of 0.0789 on the tail classes, while remaining stable, but unsatisfactory on the rest. The Class-balanced loss with the highest accuracy of 0.3787 across all test datasets has a performance of negligence.

Comparison to Benchmark

TODO: No benchmark for ResNet50V2 trained on CIFAR-100. Closest architecture is ResNet50 from the paper *ResNet strikes back: An improved training procedure in timm* [37].
Top 1 accuracy trained on a balanced CIFAR100: Their: 86.9 %. Mine: 83.2 %.
TODO: Describe differences.

5.2.3 ViT-B/16

Results from Balanced Training Dataset

Table 5.5: Evaluation results for ViT-B/16 trained on the custom balanced dataset, showing Acc1.

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**

From table 5.5 it is clear that the loss design with the overall best perfomance on the ViT-B/16 architechture is the LDAM loss. However, the performance of all loss designs across all test datasets are noticably worse than the performance of the other model architectures in this experiment trained with the balanced training dataset. See tables 5.1, 5.3, and 5.7 for reference.
There is no noticably trend in overall worst performance across loss designs, as all loss function perform within range of each other on all datasets.
TODO: Consider if it make sense to calculate the standard deviation of results within datasets.

Results from Long-Tailed Training Dataset

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

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	0.1570	0.2750	0.3140	0.1361	0.0263

From table 5.6, the best perfomance on the balanced test dataset is accomplshed by th Balanced Softmax loss (Acc1: 0.2460) which also has the best perfomance on middle classes (Acc1: 0.2130), while the Softmax loss achieves the best performance on the long tailed dataset (Acc1: 0.4367) as well as the head classes (0.5071). The best performance on tail classes is achieved by the Class-Balanced loss (Acc1: 0.1053) TODO: check this again, however this loss design has an underwhelming performance across all other test sets with the lowest achieved perfomance on the head classes with 0.000 accuracy TODO: explain.
In all tests, the only loss design achieving a higher accuracy than 50 % is the Softmax loss on the head classes, meaning that the model underperforms across all loss designs.

Comparison to Benchmark

The best result from ViT-B/16 trained with CIFAR100 is obtained in *Perturbated Gradients Updating within Unit Space for Deep Learning* [36] with the best accuracy of 93.95 %. In comparison, the highest achieved accuracy on a balanced test dataset in this experiment on ViT-B/16 trained with a balanced CIFAR100 was 59.1 %. See table 5.5. TODO: Compare methods. Link to paper: `https://arxiv.org/pdf/2110.00199v2`
TODO: Make a table with results.

5.2.4 ConvNeXt Base

Results from balanced Training Dataset

From table 5.7, the best performance on the balanced test dataset is achieved by the Balanced Softmax Loss (Acc1: 0.8364), while also achieving the best result on the tail classes (Acc1: 0.9474).
Not surprisingly, the Softmax loss, Weighted Softmax loss, and Class-balanced loss exhibits the same performance across all test dataset, as they were trained

Table 5.7: Evaluation results for ConvNeXt Base trained on the custom balanced dataset, showing Acc1.

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

with the balanced training dataset, and their designs reduces to the Softmax loss. These three loss function exhibit the best performance on the long-tailed dataset (0.8535), head classes (Acc1: 0.8566) and tail classes (Acc1: 0.9474). The best performance on middle classes is achieved my the Equalization loss (Acc1: 0.8343). Five out of six loss design achieve the same performance on the tail classes, likely due to saturation causes by the limited number of samples and noise.
TODO: Explain why balanced softmax could achieve higher accuracy in the balanced dataset.

Results from Long-Tailed Training Dataset

Table 5.8: Evaluation results for ConvNeXt Basetrained on the long-tailed dataset, showing Acc1.

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

From table 5.8 the best performance on a balanced test dataset is achived by the Balanced Softmax loss (Acc1: 0.6460), which was the same for the ConvNeXt Base trained with the balanced CIFAR100. See table 5.7. Likewise, the Balanced Softmax loss performs with the highest accuracy on the tail classes (Acc1: 0.5789), far exceeding the second highest performance, achieved by Equalization loss (Acc1: 0.3421). The highest accuracy on the long-tailed dataset (Acc1: 0.8316), as well as the head classes (Acc1: 0.8898), is achieved by the Softmax Loss, while Equalization Loss performs with highest accuracy on middle classes (Acc1: 0.6923).

In comparison, the Balanced Softmax loss is third in accuracy on the long-tailed dataset, head classes, as well as middle classes.
Noticably, both Softmax loss and focal loss performs with equal accuracies on both middle (Acc1: 0.6568) and tail classes (Acc1: 0.3158), but not elsewhere. TODO: give a reason why that might be.
The worst performance is achieved by the Class-Balanced loss, with accuracies far below the second worst performances. The Class-balanced loss should be disregarded for training with a long-tailed dataset, as it is most likely a fault in implementation. TODO: investigate this and return to this conclusion.

Comparison to Benchmark

TODO: No benchmark found for ConvNeXt Base trained with CIFAR100. Closest is *Conv2NeXt: Reconsidering Conv NeXt Network Design for Image Recognition* by Feng et al. with top 1 accuracy of 83.82 % in CIFAR-100. Mine is 83.64 %. Link: `https://ieeexplore.ieee.org/document/10072172`

5.3 Comparison of Models

The model are compared by taking the mean of the results of the loss functions for each model. The mean and standard deviation for each model are shown on figure 5.7, and on figure 5.8 excluding the Class-Balanced Loss.

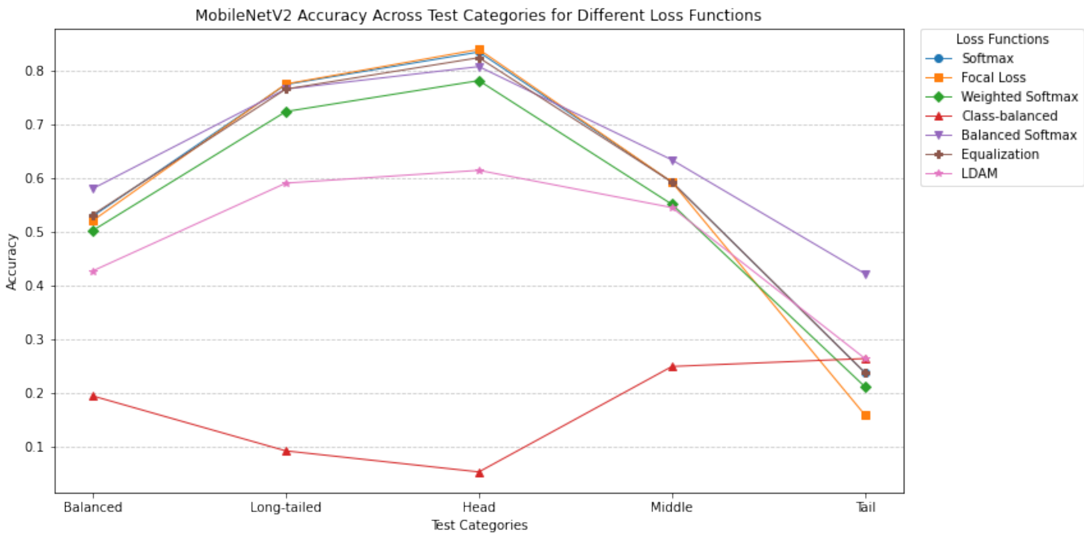

Figure 5.3: MobileNetV2 top 1 accuracy across test categories (Balanced, Long-tailed, Head, Middle, Tail) for different loss functions.

Mean and Standard Deviation

TODO: Make a table for the mean and standard deviations.

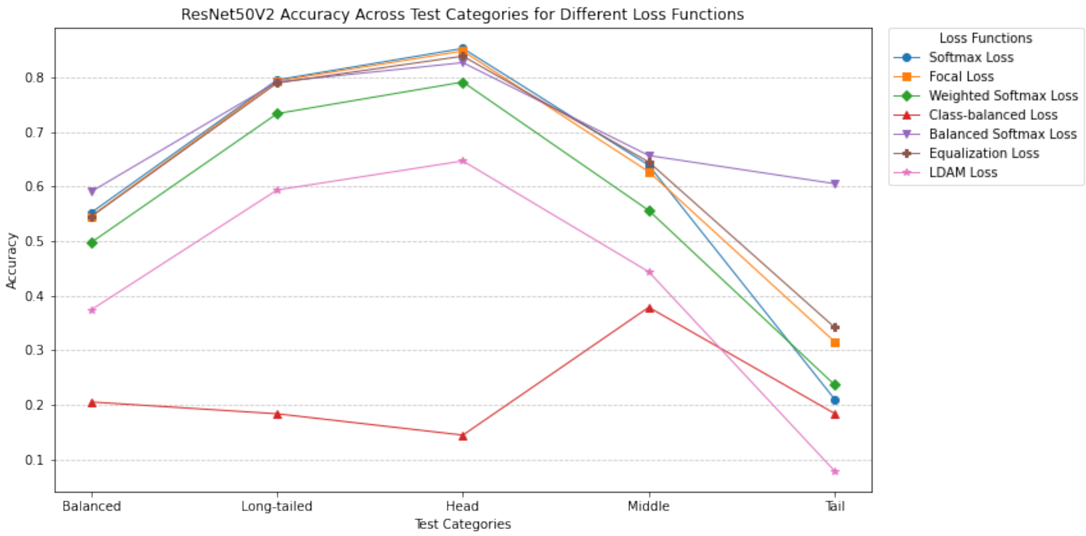

Figure 5.4: ResNet50V2 top 1 accuracy across test categories (Balanced, Long-tailed, Head, Middle, Tail) for different loss functions.

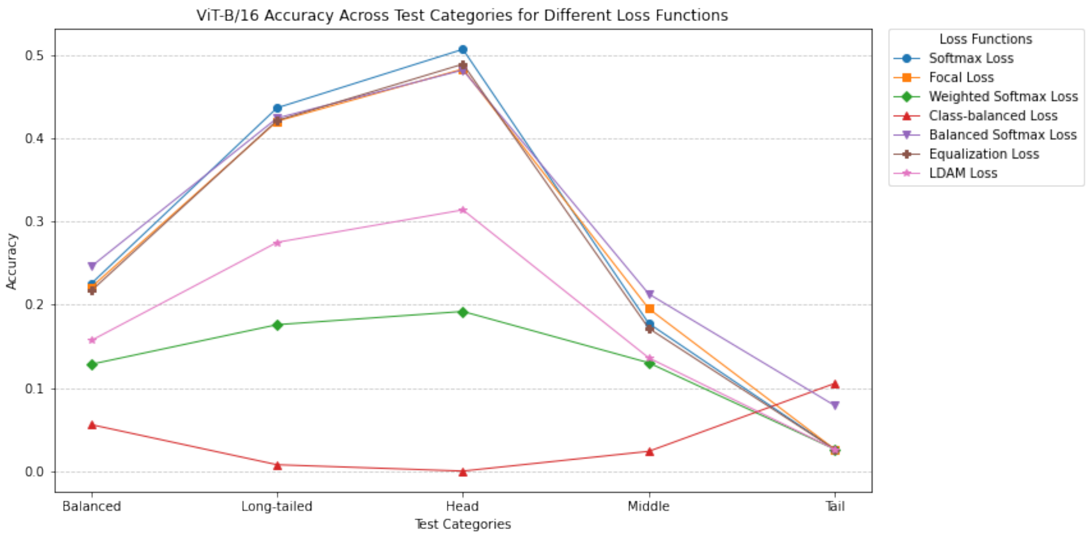

Figure 5.5: ViT-B/16 top 1 accuracy across test categories (Balanced, Long-tailed, Head, Middle, Tail) for different loss functions.

The plots on figures 5.7 and 5.8 compares the performance of the models MobileNetV2, ResNet50V2, ViT-B/16, and ConvNeXt Base across the evaluation categories: balanced, long-tailed, head, middle, and tail. Each model is slightly offset along the x-axis to avoid overlap. The mean and standard deviation in figure 5.7 include the Class-Balanced Loss while figure 5.8 does not.

Each point represents the mean accuracy of a model for a specific category and the bars represent the standard deviation. A model with higher mean accuracy is performing better in that category, and a smoother or higher line indicates consistent performance across all categories.

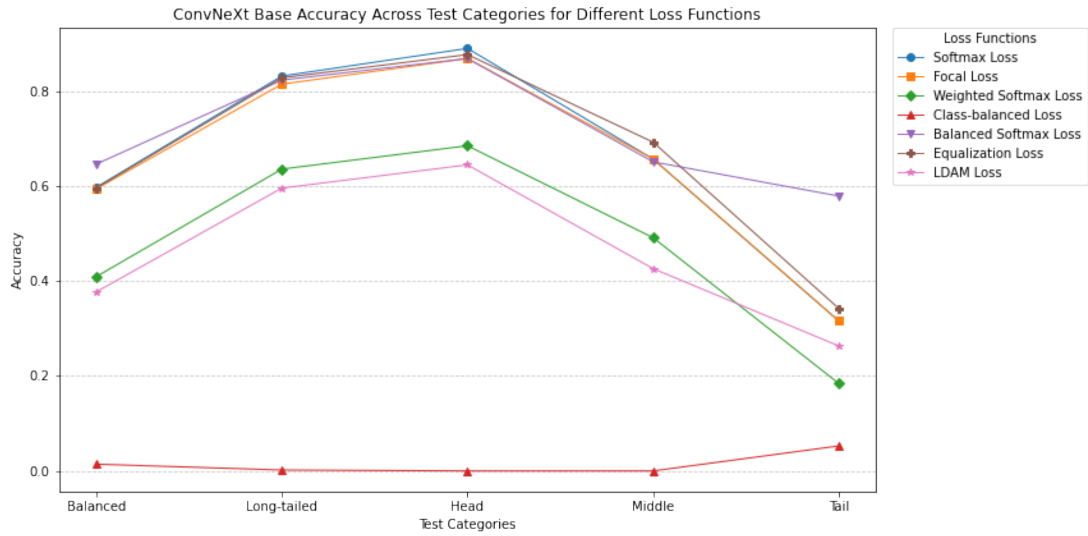

Figure 5.6: ConvNeXt Base top 1 accuracy across test categories (Balanced, Long-tailed, Head, Middle, Tail) for different loss functions.

The error bars show the variability in accuracy across different loss functions for each model and category. A longer error bar means the performance is less consistent and the performance is more dependent on the chosen loss function, while shorter error bars indicate consistant performance.
The plot in figure 5.8 shows that the model with the best average performance on tail classes is the ConvNeXt Base architecture, while also showing a strong performance in other categories.
MobileNetV2 is the least dependent on loss functions, exhibiting the smallest standard deviation, while the ResNet50V2 architecture, compared with figures 5.4 and 5.6, is the most dependent on loss functions when exhibiting tail performance, as the Balanced Softmax Loss outperforms the remaining loss functions, as seen in figure 5.4 and table 5.4.
The model exhibiting the worst performance is the ViT-B/16 with a significant lower mean than the other three models. TODO: reference to figures and tables. TODO: Comparison of overall model performance. Potentially statistical analysis, e.g. ANOVA, Tukey's HSD.

5.4 Comparison of Loss Functions

TODO: Comment on the lack of results from the Class-Balanced Loss on all sets except the tail classes.
From figure 5.9 it can be seen that the Balanced Softmax Loss has a better average perfomance on tail classes, while also displaying the smoothest variation across evaluation categories, ignoring the Class-Balanced loss. However, the standard deviation suggests that the perfomance of the Balanced Softmax Loss depend on the model architecture.

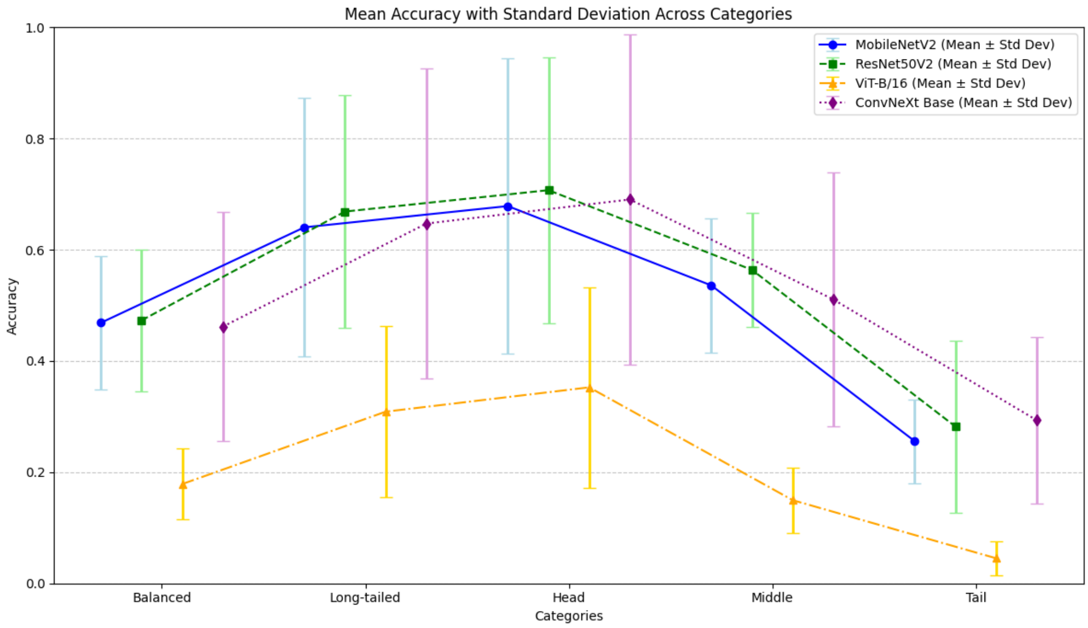

Figure 5.7: Mean Accuracy with Standard Deviation Across Categories for MobileNetV2, ResNet50V2, ViT-B/16, and ConvNeXt Base.

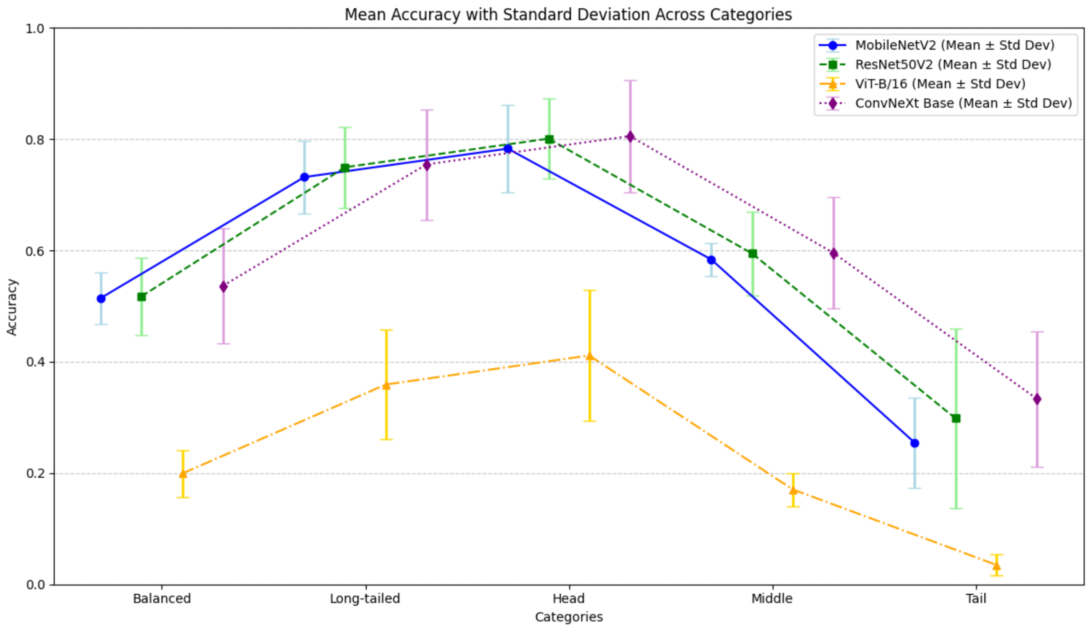

Figure 5.8: Mean Accuracy with Standard Deviation Across Categories for MobileNetV2, ResNet50V2, ViT-B/16, and ConvNeXt Base without Class-Balanced Loss.

5.5 Comparison with Benchmarks

The results are compared to the best published results for MobileNetV2, ResNet50V2, ViT-B/16 and ConvNeXt Base on CIFAR-100.

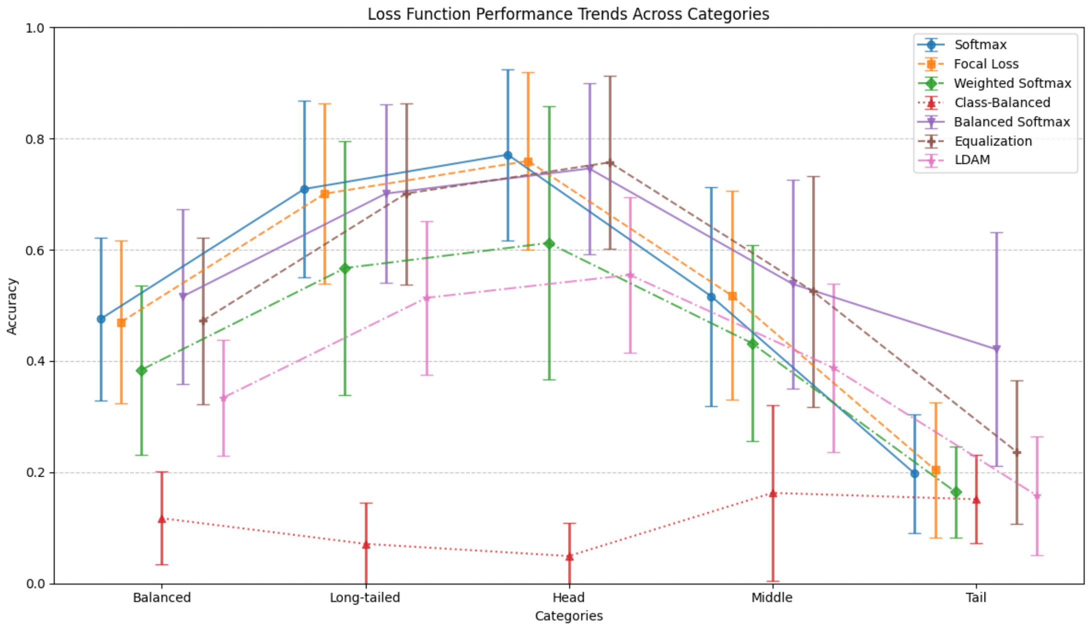

Figure 5.9: Performance trends of different loss functions across evaluation categories. Error bars indicate the standard deviation of accuracy across models, highlighting variability in performance for each loss function.

To contextualize the performance of models, the results are compared against the best posted results for the same architecture on CIFAR-100, reported by Author/Source.
TODO: Move benchmark results into a table.

5.6 Summary and Discussion

Things for discussion:

- Why is the performance on tail classes better than the performance on head and middle classes for the balanced training data, and what could it mean for the results on the long-tail training data?
- Running multiple of the same training to check for variance. What could this tell us in terms of statistical significance?
- The LDAM should be run with DRW as well. Compare to articles that run both LDAM and LDAM-DRW.
- Including or omitting the Class-Balanced Loss.
- Discuss why the ViT-B/16 model underperforms but shows excellent performance in other studies.

The Balanced Softmax Loss proved to be the most effective loss function for tail classes, consistently achieving the highest accuracy on these classes across all models while also achieving competing results on other datasets. This results highlight its strenght in re-weighting logits to account for imbalanced datasets, particular for tail classes, where the Softmax baseline loss function falter. The synergy between Balanced Softmax and certain architectures, such as ConvNeXt Base, was espicially notable, suggesting that specific combinations of models and loss functions can achieve superior performance.

However, this improvement in tail-class accuracy often came with trade-offs. For example, while ResNet50V2 achieved a top 1 accuracy of 0.6053 on tail classes using Balanced Softmax, its head-class accuracy (0.8270) lagged behind ConvNeXt Base (0.8685). This trade-off highlights the challenge of optimizing for both head and tail classes simultaneously and indicates that Balanced Softmax prioritizes tail-class adjustments at the expense of head-class performance. TODO: mention statistical significance.

Class-Balanced Loss, in contrast, consistently underperformed across all models and datasets. This discrepancy between its intended purpose and observed results suggests possible issues with its weighting strategy or implementation. Further investigation is necessary to understand these limitations and determine whether its performance can be improved.

Notably, the ViT-B/16 architecture underperformed significantly across all loss functions and datasets, with an the best accuracy of 59.06% on a balanced training dataset compared to its benchmark of 93.95%. This suggests that Vision Transformers, despite their reported success in other contexts, may require more data, longer training, or additional fine-tuning. The results indicate that the default ViT-B/16 architecture may not be well-suited for long-tailed datasets without further optimization.

While Balanced Softmax stood out as the most robust loss function overall, statistical validation is required to confirm the significance of these findings, particularly when comparing closely performing configurations.

Overall, these findings emphasize the importance of aligning loss functions with both dataset characteristics and model architectures to address the challenges of deep 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] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. "Multilayer feedforward networks are universal approximators". In: *Neural Networks* 2.5 (1989), pp. 359–366. ISSN: 0893-6080. DOI: https://doi.org/10.1016/0893-6080(89)90020-8. URL: https://www.sciencedirect.com/science/article/pii/0893608089900208.

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

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

[21] Y. LeCun et al. "Backpropagation Applied to Handwritten Zip Code Recognition". In: *Neural Computation* 1.4 (1989), pp. 541–551. DOI: 10.1162/neco.1989.1.4.541.

[22] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems*. Ed. by F. Pereira et al. Vol. 25. Curran Associates, Inc., 2012. URL: https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[23] Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2015. arXiv: 1409.1556 [cs.CV]. URL: https://arxiv.org/abs/1409.1556.

[24] Christian Szegedy et al. *Going Deeper with Convolutions*. 2014. arXiv: 1409.4842 [cs.CV]. URL: https://arxiv.org/abs/1409.4842.

[25] Kaiming He et al. *Identity Mappings in Deep Residual Networks*. 2016. arXiv: 1603.05027 [cs.CV]. URL: https://arxiv.org/abs/1603.05027.

[26] Andrew G. Howard et al. *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*. 2017. arXiv: 1704.04861 [cs.CV]. URL: https://arxiv.org/abs/1704.04861.

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

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

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

[30] 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.

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

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

[33] Hugging Face. *ViT Base Patch16-224 by Google*. https://huggingface.co/google/vit-base-patch16-224. Accessed: 2024-12-04. 2024.

[34] PyTorch Team. *ConvNeXt Base Model*. https://pytorch.org/vision/main/models/generated/torchvision.models.convnext_base.html. Accessed: 2024-12-04. 2024.

[35] PyTorch Team. *Vision Transformer (ViT-B-16)*. https://pytorch.org/vision/main/models/generated/torchvision.models.vit_b_16.html. Accessed: 2024-12-04. 2024.

[36] Ching-Hsun Tseng et al. "Perturbed Gradients Updating within Unit Space for Deep Learning". In: *2022 International Joint Conference on Neural Networks (IJCNN)*. IEEE, July 2022, pp. 01–08. DOI: 10.1109/ijcnn55064.2022.9892245. URL: http://dx.doi.org/10.1109/IJCNN55064.2022.9892245.

[37] 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	0.1570	0.2750	0.3140	0.1361	0.0263

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	43.5990	0.1570	0.1321	17.1295	0.2750	0.2769	11.5903	0.3140	0.3466	30.5370	0.1361	0.1791	80.9656	0.0263	0.0351

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.