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

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.

Acknowledgements

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

Contents

Chapter 1

Introduction

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 Related Work

The challenge of long-tailed datasets has been extensively studied in the literature [1, 2].

Data Re-sampling Data re-sampling techniques aim to modify the training dataset to mitigate class imbalance by simulating a balanced dataset. The most popular methods for re-sampling are random over-sampling (ROS) and random under-sampling (RUS) [3, 4]. ROS involves duplicating random samples from the minority classes, whereas RUS involves removing samples from the majority classes [1, 4]. However, these techniques have their weaknesses: over-sampling the minority classes will eventually lead to overfitting tail classes, and under-sampling can lead to underperformance on head classes [1]. Another re-sampling technique, called Synthetic Minority Over-Sampling Technique (SMOTE) [3], involves synthetical generation of instances of the minority classes based on the existing data.

Transfer Learning Transfer learning techniques adress the issue of class imblance by transferring learned features from head classes to tail classes. Examples include transferring the intra-class variance [5] and transferring semantic deep features [6].

Data Augmentation Data augmentation is a deep learning technique used to expand the training dataset by applying transformations to samples or features, enhancing model accuracy while reducing overfitting [7, 8]. Common augmentation methods include Mixup [9], which combines pairs of samples to create interpolated examples; CutMix [10], which replaces regions of one image with patches from another; and UniMix [11], which focuses on calibrating the feature space for long-tailed distributions. Rare-class Sample Generator (RSG) [12] focuses on enhancing tail-class performance by transferring knowledge from head classes.

Decoupled Training Decoupled training [13] separates representation learning and classifier training. First, the model is trained to extract features from the entire dataset, afterwards, a re-balancing technique is applied during classification. Despite the simple framework, decoupling showed state-of-the art performance on long-tailed benchmarks.

Ensemple Learning Ensemble learning is a technique that combines multiple network modules (i.e. experts) to solve long-tailed problems. By doing so, the predictions of the different models can be combined, ultimately outperforming a single model [14, 15].

1.3 Reading Guide

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

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 [16, 6]. For example, the iNaturalist, a popular benchmark for image classification, exhibits a long-tailed distribution of species [17]. Other benchmarks are constructed by sampling from datasets such as ImageNet [18] by using a Pareto distribution, which simulates long-tailed class distributions with a power-law decay [1, 19, 20].

CIFAR100-LT [20], derived from the CIFAR-100 dataset [21], 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 an exponential decay en sample sizes, given by:

$$n_i = n_{max} \cdot \mathrm{IR}^{\frac{i-1}{C-1}} \tag{2.1}$$

Where n_i is the number of samples in class i, n_{max} is the number of samples in the most frequent class, IR is the imbalance ratio, and C is the total number of classes [22].

Other long-tailed datasets follow a Pareto distribution with number of samplers per class as followed [6]:

$$f(C) = \frac{\alpha C_{\mathrm{m}}^{\alpha}}{C^{\alpha+1}}, \quad C \geq C_{\mathrm{m}}, \quad \alpha > 0 \tag{2.2}$$

Here, α is the shape parameter, C_{m} is the scale parameter representing the minimum possible class index, and C is the class index. A larger α results in a more severe imbalance.

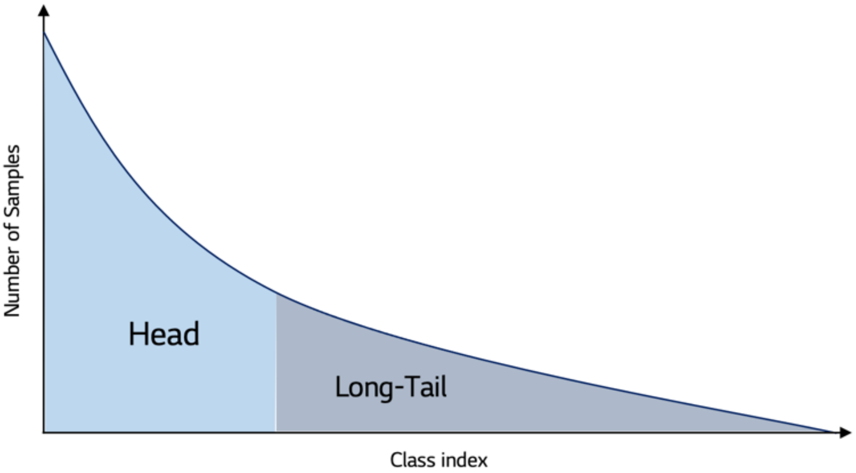

Figure 2.1: Illustration of a long-tailed distribution. Figure from [23].

Class imbalance has a profound impact on model performance compared to evenly distributed datasets [24, 25]. 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 [26], ResNet50V2 [27], and ConvNeXt Base [28] as the CNN architectures, and ViT-B/16 [29] 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 [30, 31].
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.

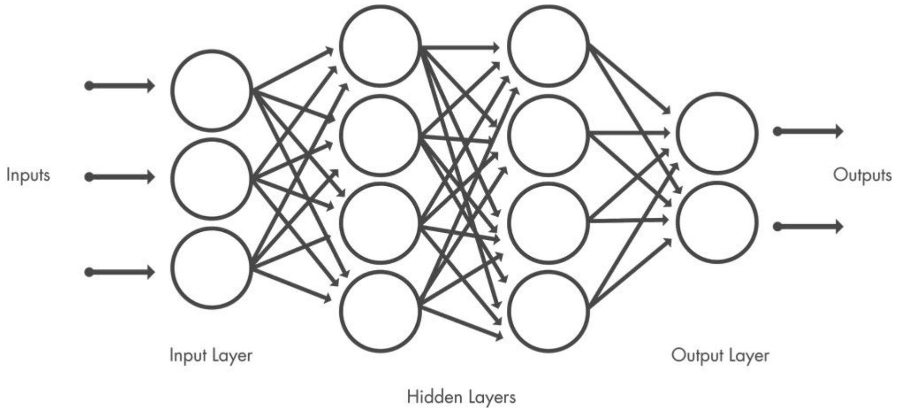

Figure 2.2: Layers of a neural network. Figure from [32]. TODO: Make this figure.

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 [30].
TODO: Include how the weights are updated with gradients based on the loss function.

2.2.2 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) [33] 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 [34, 35, 30].
CNNs consist of several core components, as illustrated in Figure 2.3. They have three main layers: convolutional layers, pooling layers, and a fully connected layer. The convolutional layer is the first layer, and serves to 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. The CNN can be made up of multiple convolution and pooling layers, but the fully connected layer will be the final layer. 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.

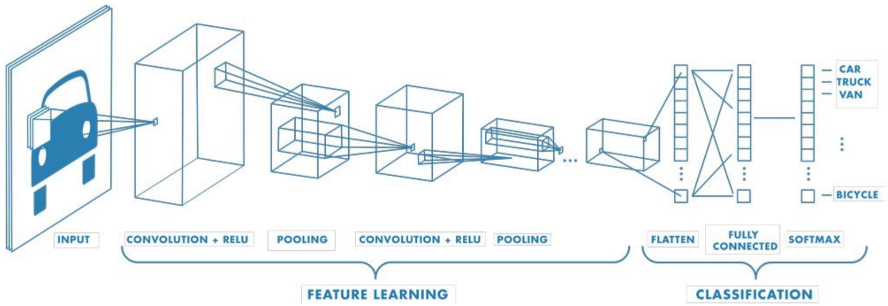

Figure 2.3: Illustration of a convolutional neural network. Figure from [32]. TODO: Make this figure.

TODO: Tie to the models used for experiments in this thesis. `https://www.ibm.com/topics/convolutional-neural-networks`
CNNs gained popularity after the introduction of LeNet-5 by LeCun et al. in 1998 [34] that demonstrated the potential of CNNs by recognizing handwritten digits. Later, AlexNet [35] 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 [36], GoogLeNet [37], and ResNet [27], which have set the stage for the advancements seen in modern CNNs.

ResNet50 Architecture

TODO: write this section. ResNet50 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 [27].

MobileNetV2 Architecture

MobileNetV2 introduced by Sandler et al. [26] 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 [38], 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) [27] TODO: as discussed in the previous section, allow for identity mapping and improved gradient flow, MobileNetV2 employs an inverted residual structure [26]. 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.

Depthwise Separable Convolution Following MobileNetV1, MobileNetV2 relies on depthwise separable convolutions to factorize the convolution operation into two simpler operations [38]: 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 [38, 26].

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

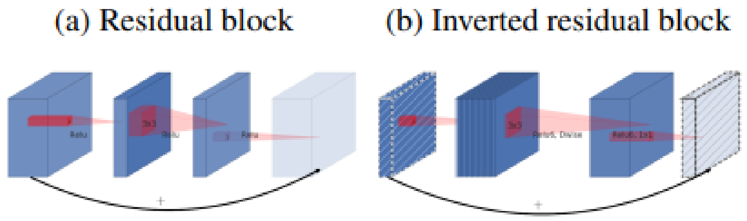

Figure 3: The difference between residual block [8, 30] and inverted residual. Diagonally hatched layers do not use non-linearities. We use thickness of each block to indicate its relative number of channels. Note how classical residuals connects the layers with high number of channels, whereas the inverted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

Figure 2.4: Residual and Inverted Residual Blocks. Figure from [26]. TODO: Make this figure.

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 [28]. 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 Trans-

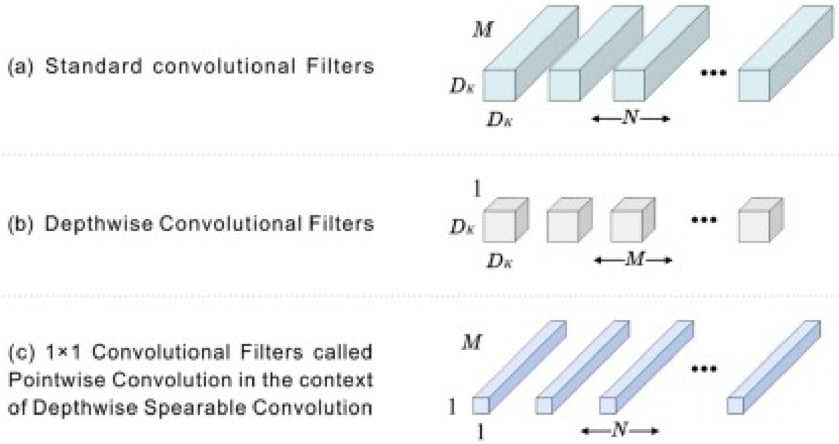
(a) Standard convolutional Filters
M
D_K
D_K
N
(b) Depthwise Convolutional Filters
1
D_K
D_K
M
(c) 1×1 Convolutional Filters called Pointwise Convolution in the context of Depthwise Spearable Convolution
M
1
1
N

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

formers, 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 [29]. 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 ap-

proach of adding an extra learnable "classification token" to the sequence is used. `https://paperswithcode.com/method/vision-transformer`

2.3 Classic Long-Tailed Methods

TODO: 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.

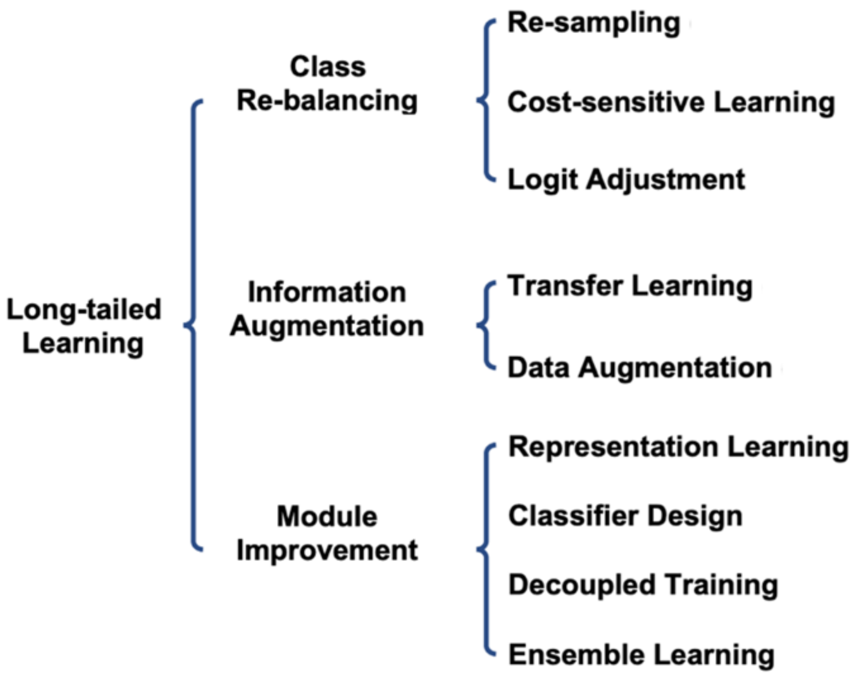

Figure 2.6: Long-tailed categories as described by *Zhang et al.* [1].

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].
TODO: Include little about re-sampling, and focus on class-sensitive learning and an little on logit adjustment.

Re-sampling

The traditional way to sample when training deep networks is using 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.

TODO: Rewrite this section to closer resemble the related work.

Class-sensitive Learning

Traditional training methods using the standard loss function, cross-entropy loss, can lead the model to be biased towards head classes, as the loss ignores class imbalance and thus generate an uneven amount of gradients for different classes [1]. Consequently, tail classes are often misclassified. To address this imbalance, class-sensitive learning methods modify the loss function to pay more attention to minority classes, thus improving overall performance.

Two prominent categories of class-sensitive strategies are *re-weighting* and *re-margining*. Re-weighting involves adjusting the contribution of each class to the loss by multiplying them with different weights. By carefully selecting these weights, the model allocates a higher penalty to misclassification of underrepresented classes, thus re-balancing the training process [1]. Re-margining, on the other hand, modifies the decision boundaries by introducing class-dependent margins, ensuring that minority classes have more room to establish discriminative features in the embedding space [1, 20].

In what follows, the theory behind several representative loss functions commonly used in class-sensitive learning is presented, beginning with an introduction to loss functions followed by the baseline Softmax Cross-Entropy loss. Subsequently, the theory behind multiple re-weighting schemes including Weighted Softmax Cross-Entropy, Focal Loss, Class-Balanced Loss, Balanced Softmax Loss, and Equalization Loss, as well as the re-margining method LDAM Loss. Each of these methods seeks to improve the imbalance in training through modifications that ultimately improve the classification performance on tail classes.

Table 2.1 summarizes the loss functions employed for class-sensitive learning used in this thesis, outlining their formulations and categorizations into re-weighting or re-margining strategies.

Table 2.1: Summary of losses.

Loss Name	Formulation	Type
Softmax loss [39]	$\mathcal{L}_{ce} = -\log(p_y)$	-
Focal loss [40]	$\mathcal{L}_{fl} = -(1-p_y)^\gamma \log(p_y)$	re-weighting
Weighted Softmax loss [1]	$\mathcal{L}_{wce} = -\frac{1}{\pi_y}\log(p_y)$	re-weighting
Class-balanced loss [25]	$\mathcal{L}_{cb} = -\frac{1-\gamma}{1-\gamma^{n_y}}\log(p_y)$	re-weighting
Balanced Softmax loss [41]	$\mathcal{L}_{bs} = -\log\left(\frac{\pi_y \exp(z_y)}{\sum_j \pi_j \exp(z_j)}\right)$	re-weighting
Equalization loss [42]	$\mathcal{L}_{eq} = -\log\left(\frac{\exp(z_y)}{\sum_j \omega_j \exp(z_j)}\right)$	re-weighting
LDAM loss [20]	$\mathcal{L}_{ldam} = -\log\left(\frac{\exp(z_y-\Delta_y)}{\sum_j \exp(z_j-\Delta_j)}\right)$	re-margining

Introduction to Loss Functions Loss functions are a fundamental component in machine learning and deep learning [30, 43]. They serve as a measure of how far a model's predictions are from the actual values, where higher loss indicates greater prediction error, whereas a lower loss implies more accurate predictions. Thus, the goal of training a model is to minimize the loss.

The loss function plays a crucial role, as it guides the model to adjust its parameters (weights and biases) through an optimization algorithm [43]. The algorithm calculates the gradient of the loss function with respect to each parameter to determine how the parameter affect the loss. Afterwards, the model update its parameter to reduce the loss, following:

$$w_{t+1} = w_t - \eta \frac{\partial L}{\partial w_t} \tag{2.3}$$

Here, w are the parameters, η is the learning rate, and L is the loss function. This is an iterative process, and continues until the loss reaches a minimum or the model converges.

From equation (2.3), the choice of loss function is critical as it influences the model's interpretation of prediction errors and parameter adjustment. For image classification tasks, the standard loss function is the cross-entropy loss combined with the softmax activation function [30] described in the following.

Softmax Cross-Entropy Loss The *Softmax Cross-Entropy (CE) loss* is a fundamental building block in training deep classifiers and is widely regarded as the baseline in classification tasks [1, 44, 39]. The softmax loss, as it is commonly known as, is the combination of the cross-entropy loss and the softmax activation function [30, 43]. First, the need of the softmax activation function will be decribed.

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:

$$P(y = i \mid \mathbf{z}) = \frac{\exp(z_i)}{\sum_{j=1}^{K} \exp(z_j)} \tag{2.4}$$

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:

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

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

$$\mathcal{L}_{\text{CE}} = -\log(p_y) \tag{2.6}$$

This equation 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.
While this approach provides a robust and stable training objective, it inherently treats all classes equally and does not account for class imbalance.

Weighted Softmax Cross-Entropy Loss The *Weighted Softmax Cross-Entropy (WCE) loss* modifies the standard CE loss (equation (2.6)) by introducing a weighting factor w. For multiclass classification, the CE loss it is multiplied by the inverse class frequencies $1/\pi_y$ [1, 40], given by:

$$\mathcal{L}_{\text{WCE}} = -w \log(p_y) = -\frac{1}{\pi_y} \log(p_y) \tag{2.7}$$

This weighted CE loss ensures that minority classes contribute more to the overall loss, addressing the imbalance during training.

Focal Loss *Focal Loss* [40] addresses class imbalance by improving model performance on hard-to-classify examples by focusing on wrongly classified examples. The technique for this is called down-weighting, and uses the prediction probabilities to dynamically adjust the contribution of each sample to the loss. Well-classified examples with high probabilities p_y are down-weighted by adding a modulating factor $(1 - p_y)^\gamma$ to the cross-entropy loss (eq. (2.6)). Here, $\gamma \geq 0$ is a tunable focusing parameter. The Focal Loss modifies the CE loss by applying the inverse prediction probability as follows:

$$L_{fl} = -(1 - p_y)^\gamma \log(p_y) \tag{2.8}$$

This factor increases the weight of misclassified examples, ensuring the model prioritizes learning from challenging samples. Figure 2.7 shows how the focal loss for different values of γ. When $\gamma = 0$ the modulating factor equals 1, and the focal loss becomes the standard CE loss. The blue line in figure 2.7 represents the standard cross-entropy loss for $\gamma = 0$.

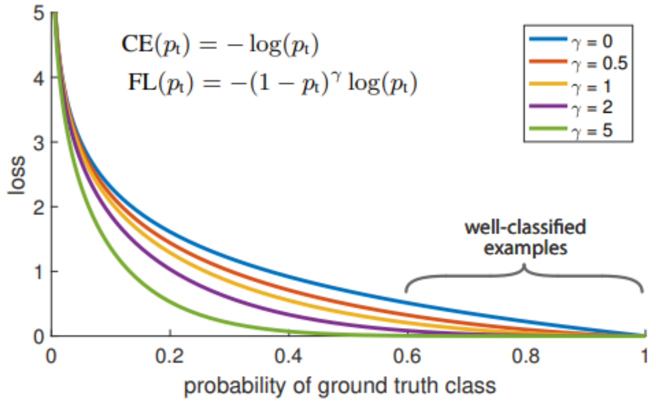

Figure 2.7: Figure from [40]. TODO: Make this figure

Class-Balanced Loss *Class-Balanced (CB) Loss* [25] introduces a re-weighting strategy based on the effective number of samples per class to re-balance the loss. Instead of simply using raw class frequencies, CB Loss estimates how much additional information new samples provide, acknowledging an information overlap among data, and as the number of samples increases, the marginal benefit of extracted features from new data diminishes. As illustrated in figure 2.8, the feature space is the map of all possible data, and each sample occupies a part of the space. Collecting more samples of a class means that there is a probability that their features overlap, meaning that additional samples diminishes new information. In feature space, the probability of newly sampled data with volume 1 is overlapping p, which means that the probability of adding new information to the feature space is $(1-p)$.

Cui et al. [25] introduced the effective number of samples as the expected volume of samples, denoted E_n, where $n \in \mathbb{Z}_{>0}$ is the number of sampels. The effective number of samples is defined as:

$$E_n = \frac{1-\beta^n}{1-\beta} \tag{2.9}$$

where $\beta = (N-1)/N$. The hyperparameter $\beta \in [0,1)$ determines how fast E_n grows.

As a result, the CB loss introduces a class-balanced re-weighting term, inversely proportional to the effective number of classes. Adding the CB term to the standard softmax cross-entropy given in equation (2.6) yield the following:

$$L_{cb} = -\frac{1-\beta}{1-\beta^{n_y}}\log(p_y) \tag{2.10}$$

where n_y is the number of samples in the ground truth class y. Applying the inverse of the effective number ensures that minority classes contribute more to

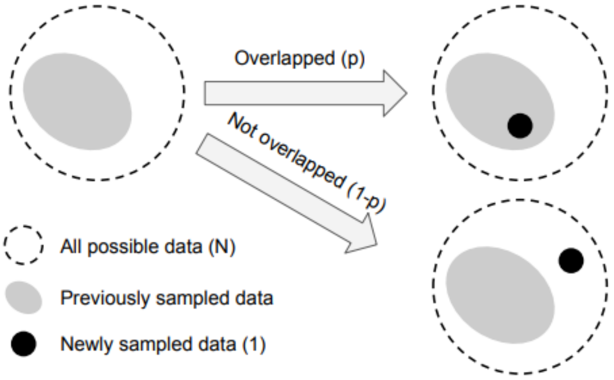

Figure 2. Giving the set of all possible data with volume N and the set of previously sampled data, a new sample with volume 1 has the probability of p being overlapped with previous data and the probability of $1 - p$ not being overlapped.

Figure 2.8: Figure from [25]. TODO: Make this figure

the total loss, as the effective number in equation (2.9) grows large for minority classes (small n_y) and small for majority classes (large n_y). When $\beta = 0$ the loss is equivalent to the CE loss. Contrary, when $\beta \longrightarrow 1$ the re-weighting effect grows.

Equalization Loss *Equalization (EQ) Loss* [42] aims to mitigate the over-suppression of tail classes, which occurs when these underrepresented classes serve predominantly as negative examples for the more frequent classes. The idea is to ignore the gradients for rare classes so that they are not excessively penalized when they appear as negatives, preventing their learned representations from being overshadowed.
The EQ loss was designed for object recognition, but the authors decided to adopt the principles into image classification, officially called the *Softmax Equalization (SEQ) Loss*, defined as:

$$\mathcal{L}_{SEQL} = -\sum_{j=1}^{C} y_j \log(\tilde{p}_j) \tag{2.11}$$

where

$$\tilde{p}_j = \frac{e^{z_j}}{\sum_{k=1}^{C} \tilde{w}_k e^{z_k}} \tag{2.12}$$

and the weight term is given by:

$$\tilde{w}_k = (1 - \beta T_\lambda(f_k))(1 - y_k) \tag{2.13}$$

Here, y_j is the ground truth class, z_j is the logit for class j, f_k is the frequency of class k, $T_\lambda(\cdot)$ is the threshold on class frequency, and β is a random variable taking value 1 with probability γ and 0 with probability $1 - \gamma$.

Logit Adjustment

Logit adjustment is a class re-balancing technique that aims to optimize the class imbalance by adjusting the prediction logits [45], typically based on prior class probabilities.

Balanced Softmax Loss The *Balanced Softmax (BS)* Loss modifies the standard softmax formulation by directly incorporating class priors π_y into the logits before computing probabilities [41]. Unlike approaches that operate solely in the loss space, BS Loss integrates class frequency adjustments at the probability computation stage, effectively neutralizing the bias introduced by imbalanced class distributions. The BS loss function is as follows:

$$L_{bs} = -\log \left(\frac{\pi_y \exp(z_y)}{\sum_j \pi_j \exp(z_j)} \right) \tag{2.14}$$

where z_y represents the logit for the true class, π_y is the class prior (e.g. normalized frequency of samples from class y), and the term in the denominator normalizes the probabilities across all classes while accounting for priors.

Re-margining

Re-margining aims to solve class imbalance by improving classification by introducing margin between classes, encouraging higher margins between rare classes. TODO: Describe margins in detail, and how they impact long-tailed learning.

LDAM Loss The *Label-Distribution-Aware Margin (LDAM)* Loss represents a class-sensitive approach based on re-margining rather than re-weighting [20]. Instead of altering the loss directly, LDAM modifies the margin applied to each class' decision boundary, ensuring that minority classes have more "space" in the representation space to reduce misclassification.
"First, we propose a theoretically-principled label-distribution-aware margin (LDAM) loss motivated by minimizing a margin-based generalization bound. This loss replaces the standard cross-entropy objective during training and can be applied with prior strategies for training with class-imbalance such as re-weighting or re-sampling."

$$L_{ldam} = -\log \left(\frac{\exp(z_y - \Delta_y)}{\sum_j \exp(z_j - \Delta_j)} \right) \tag{2.15}$$

From [1]: To handle the class imbalance, re-margining attempts to adjust losses by subtracting different margin factors for different classes, so that they have a different minimal margin (i.e., distance) between features and the classifier. Directly using existing soft margin losses [138], [139] is unfeasible, since they ignore the issue of class imbalance. To address this, Label-Distribution-Aware Margin (LDAM) [18] enforces class-dependent margin factors for different classes based on their training label frequencies, which encourages tail classes to have larger margins.

Chapter 3

Methodology

This chapter describes the methods and approaches used in the experiments. This includes the selection of dataset, models, loss functions, and evaluation strategies.

3.1 Overview of Approach

Image classification involves predicting the correct category of an image from a set of predefined classes. This project focuses on addressing the challenge of class imbalance, where some classes have many samples while others have very few.

The CIFAR-100 dataset [21] was chosen because it is small enough to work with efficiently and is commonly used in other research. To create a long-tailed version of the dataset, its class distribution was adjusted to match the distribution of the ImageNet-LT dataset used in [1]. This adjustment ensures that the experiments simulate real-world class imbalance.

Several loss functions designed to handle class imbalance were tested in this project, including Softmax Cross-Entropy (CE), Focal Loss (FL), Class-Balanced (CB) Loss, Balanced Softmax (BS) Loss, Equalization (EQ) Loss, and LDAM Loss. These loss functions modify how the models learn, making it easier for them to focus on classes with fewer samples.

The experiments were performed using four model architectures: ResNet50, MobileNetV2, ConvNeXt Base, and ViT-B/16. All models were pretrained on ImageNet, which provided a strong starting point for fine-tuning on the CIFAR-100 dataset. These models were chosen because they are well-suited for image classification tasks and represent different design approaches.

Each model was trained on both a balanced training set and a long-tailed training set. The performance of the models was evaluated using a balanced test set, a long-tailed test set, and the long-tailed test set divided into head, middle, and tail classes. This division made it possible to analyze the performance for classes with different numbers of samples. The main evaluation metric used was top-1 accuracy, which measures how often the model's top prediction is correct.

In what follows, each step of the methodology will be explained in detail.

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.

3.2.1 Benchmark Dataset Selection

There are a number of benchmark datasets for long-tailed image classification tasks, including ImageNet-LT, Places365-Lt, CIFAR-100-LT, and iNaturalist 2018 TODO: references. The previous three are sampled from ImageNet, Places365, and CIFAR-100, following the Pareto distribution, as described in section 2.1, while the iNaturalist 2018 is a natural long-tailed dataset. Table 3.1 summarizes the four datasets and their data specifications.

Table 3.1: Summary of long-tailed benchmarks for image classification.

Dataset	**# Classes**	**# Training Data**	**# Test Data**
ImageNet-LT	1,000	115,846	50,000
CIFAR100-LT	100	50,000	10,000
Places-LT	365	62,500	36,500
iNaturalist 2018	8,142	437,513	24,426

With its 100 classes and only 60,000 samples, the CIFAR-100 dataset was chosen as a benchmark for the experiments conducted in this thesis based on its managable size, compared to the other benchmarks, and widespread use as a standard in image classification research TODO: reference. It's comparatively small size allows for rapid experimentation and comparisons, wheras a larger benchmark requires more computational effort and time for experiments, thus it is well-suited for multiple configurations and evaluations. However, CIFAR-100 has some limitations comparet to larger dataset, like ImageNet-LT or iNaturalist. For example, CIFAR-100 images are of relatively low resolution (32×32 pixels) compared to ImageNet (224×224 pixels), and may limit the ability of models to capture fine-grained details. Likewise, the smaller number of classes and samples can result in less diversity compared to datasets like iNaturalist 2018, which contains nearly nine times as many samples in the training data as CIFAR-100-LT. Despite these drawbacks, CIFAR-100-LT remains a practical choice for this thesis due to its computational efficiency.
TODO: Describe the type of images in CIFAR-100 and include examples.

3.2.2 Data Characteristics: Class Distribution

Understanding the class distribution in long-tailed datasets is essential for evaluating the impact of the class imbalance on model performance. This section analyzes the class distribution of the benchmark ImageNet-LT used in [1], which serves as a reference for creating a similar distribution across the CIFAR-100-LT training, evaluation, and test datasets.

The ImageNet-LT dataset, introduced by Liu et al. [6], has been widely used in empirical studies, including *Deep Long-Tailed Learning: A Survey* [1]. The class distributions of its training, validation, and test sets are shown in Figures 3.1, 3.2, and 3.3, respectively. The ImageNet-LT is constructed from the original ImageNet-2012 following the Pareto-distribution (Eq. (2.2)) with power value $\alpha = 6$ [6], resulting in an imbalance ratio of 256, meaning that the most frequent class has 256 times more samples than the least frequent class. As suggested in figure 3.1, the most frequent class has 1280 images, and the least frequent class 5 images [6]. The power value of 6 is relatively large, resulting in the steep imbalance and very few samples in the tail, as seen in figure 3.1. The underrepresentation in the tail classes poses a significant challenge for models to learn from tail classes. Both the validation and test sets are balanced, as shown in figures 3.2 and 3.3, respectively. There are 20 samples per class in the validation set, and 50 samples per class in the test set.

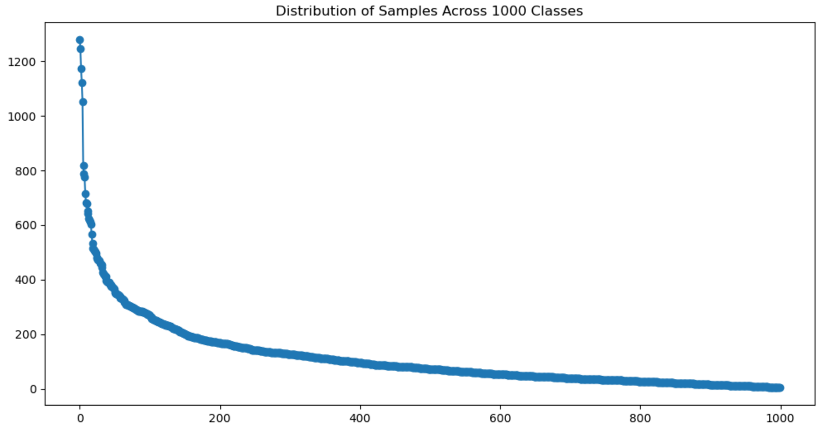

Figure 3.1: The class distribution of the training images for the ImageNet-LT dataset shows a long-tailed distribution.

However, investigating the CIFAR-100-LT dataset used in *Learning Imbalanced Datasets with Label-Distribution-Aware Margin Loss* by Cao et al. [20] shows that the training dataset follows a exponential decay, and not a pareto distribution. The imbalance factor in their studies is 100, meaning that the most frequent class has 100 times more samples than the least frequent class, following equation (2.1). This means that the imbalance is not as steep as with the Pareto distribution, resulting in a middle section of classes with fewer samples than the head classes, but more samples than the tail classes, see figure 3.4. The following section will describe the preparation of CIFAR-100 for the experiments in this thesis.

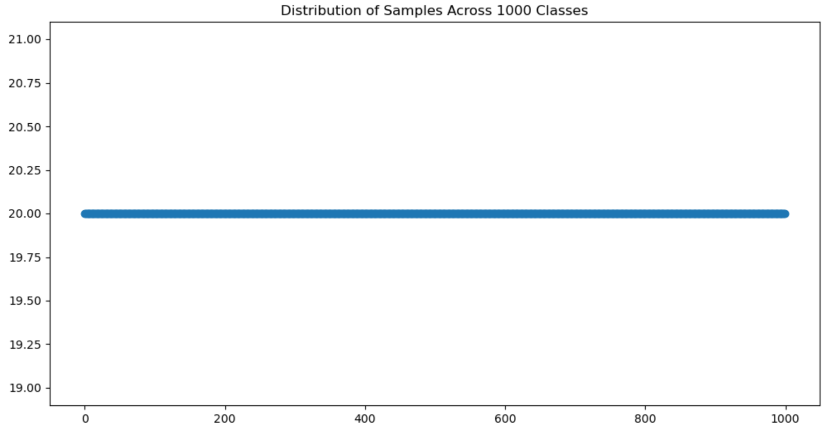

Figure 3.2: The class distribution of the validation images for the ImageNet-LT dataset shows that there are 20 samples of each class.

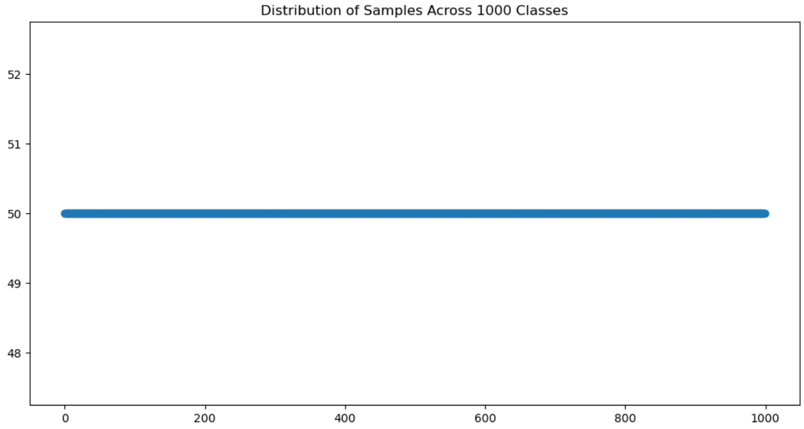

Figure 3.3: The class distribution of the test images for the ImageNet-LT dataset shows that there are 50 samples of each class.

3.2.3 CIFAR-100-LT

The experiments conducted in this thesis are trained and evaluated on the CIFAR-100 dataset. This was downloaded using the PyTorch torchvision.datasets.CIFAR100 [46]. 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.

The class distributions used in experiments in this thesis: Balanced training set, balanced test set, balanced validation set, long-tailed training set, long-tailed test set, long-tailed test set split into head, middle and tail. The class distributions can be seen in table 3.2.

To address the needs of the experiments in this thesis, the original CIFAR-100 dataset was modified to create a new split from the training data. Specifically, the

Table 3.2: Class Distributions Used in Experiments

Dataset Type	**Samples**
Balanced Training Set	45,000
Balanced Validation Set	10,000
Balanced Test Set	5,000
Long-Tailed Training Set	TODO: -
Long-Tailed Test Set	TODO: -
Head	TODO: -
Middle	TODO: -
Tail	TODO: -

training data was split into 450 samples per class for training and 50 samples per class for testing, mirroring the test data for the ImageNet. The original test set of the CIFAR-100 dataset was retained as the validation set for evaluation during training. All datasets were saved locally to maintain reproducibility.

Long-tailed Training Set: To simulate real-world scenarios with class imbalances, the training dataset was modified to introduce an exponential imbalance across the 100 classes. Following the procedure of [20], the imbalance was created using exponential decay. Here, the number of samples per class decreases exponentially, controlled by the imbalance factor, following equation (2.1). For this thesis, an imbalance factor of 0.01 was applied. This means that the most frequent class contains a 100 times more samples than the least frequent class.

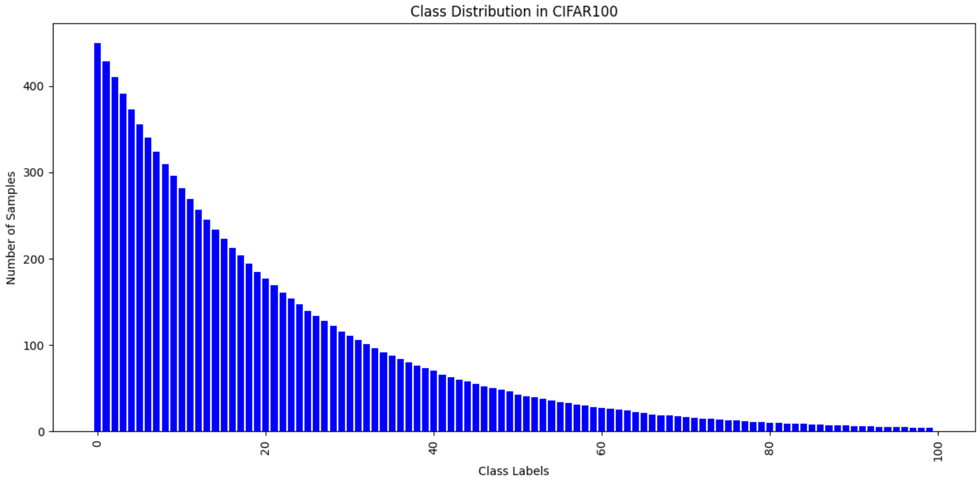

Figure 3.4: The class distribution of the CIFAR-100 long-tailed training set with 450 samples in the most frequent class.

The resulting class distribution varied from the most frequent class having 450 samples to the least frequent class having only 4 samples, as shown in figure 3.4. This imbalance ensured no class was left with zero samples, maintaining the integrity of all classes for training. The dataset was saved locally to maintain reproducibility.

Long-Tailed 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. The class distribution can be seen in figure 3.5. The dataset was saved locally to maintain reproducibility.

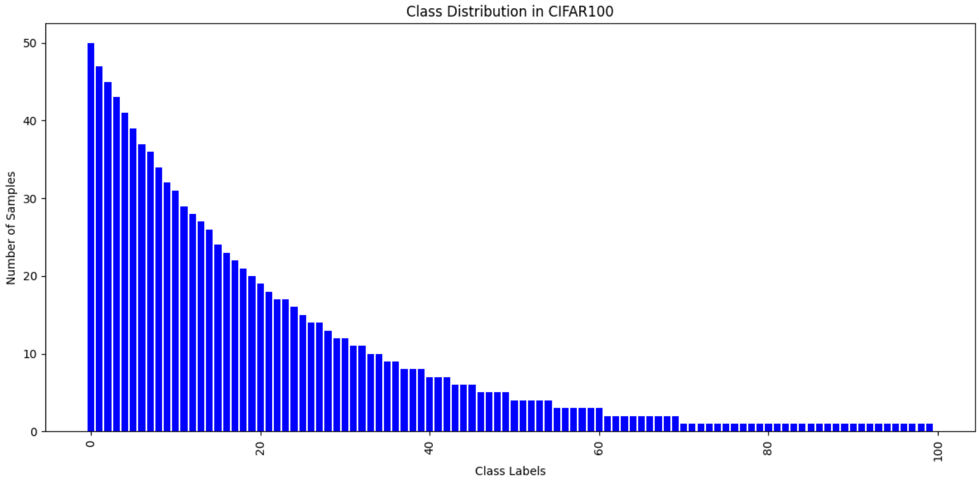

Figure 3.5: The class distribution of the CIFAR-100 long-tailed test set with 50 samples in the most frequent class.

To further analyze performance, the test set was divided into three subsets: head, middle, and tail classes. The head classes comprised the top 33% most frequent classes, the middle classes included the next 33%, and the tail classes represented the bottom 33%. See table 3.2. This categorization allows for a detailed evaluation of the methods across different classes.

3.2.4 Data Augmentation

Data augmentation and normalization were applied to the CIFAR-100 dataset to improve the generalization capability of the models. For the CIFAR-100 dataset, the augmentations were applied using the CIFAR-100 RGB statistics, with mean values [0.4914, 0.4822, 0.4465], and standard deviations [0.2023, 0.1994, 0.2010] [22]. These values are used to normalize the images after applying the transformations. However, the models used in this study are all pretrained on ImageNet, and choosing the ImageNet RGB statistics could potentially influence the training, as the pretrained weights are optimized for inputs with ImageNet statistics. Future work could explore the impact of using ImageNet statistics instead.
Images were resized to 224x224 pixels to match the input requirements of the ImageNet-pretrained models. Random cropping with a padding of 4 was applied to introduce spatial variations, while horizontal flipping with a 50% probability

further augmented the data. After these transformations, images were normalized using the CIFAR-100-specific mean and standard deviation values, ensuring consistency with the dataset.
For validation, a simpler approach was used, where images were resized to 224x224 pixels and normalized using the same mean and standard deviation values as the training set.

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.
- Write pseudo code of the implementations

Softmax Loss Without any additional weighting or margin adjustments, the standard CE loss tends to give well-represented classes an advantage. Since head classes have more training examples, each of their samples also serves as a "negative" example for all other classes, providing them with a disproportionately high number of beneficial gradients. In contrast, tail classes, having fewer positive samples, are both less represented positively and more often negatively suppressed [1, 40]. As a result, the model learns biased decision boundaries, performing well on frequent classes but poorly on rare ones. The CE loss thus serves as a natural starting point or baseline from which various class-sensitive modifications are derived. These modifications directly address the imbalance issue by altering the training dynamics, ensuring a more equitable distribution of gradients and improving the final model's performance on underrepresented classes.

Weighted Softmax Loss (1) In scenarios of severe imbalance, minority classes are often overshadowed by the abundant head classes. WCE addresses this by scaling the loss for each class according to a class-dependent factor, typically the inverse of its frequency, thereby placing a heavier penalty on misclassification of rare classes [1]. (2) By elevating the importance of tail classes in the loss, the model is incentivized to allocate more representational capacity to them, improving recall and reducing class-specific error rates. (3) Although simple, WCE provides a direct and intuitive method for re-weighting: each class's contribution to the parameter updates is modulated to reflect its scarcity or importance in the dataset distribution. (4) This helps maintain a more balanced gradient flow during training, preventing head classes from monopolizing the optimization trajectory. (5) However, one limitation is that the choice of class weights is often empirical. Naively using the inverse class frequency may not always yield optimal results, and more sophisticated weighting heuristics or data-driven methods can be employed [1]. (6) WCE sets a straightforward precedent for other re-weighting schemes, serving as a baseline for more elaborate methods that try to adapt the weights dynamically during training. (7) As it is easy to implement and understand, WCE is frequently considered as an initial step towards handling imbalance before exploring more complex techniques. (8) Overall, WCE exemplifies how re-weighting can effectively guide the model to learn less-represented categories more robustly, thereby improving long-tailed recognition performance.

Focal Loss Focal Loss modifies the CE loss by including a focusing parameter that down-weights well-classified examples, thus reducing the dominance of easily recognized classes or samples [40]. (2) By concentrating on hard examples—often those from underrepresented classes—Focal Loss ensures that the model's parameter updates focus on improving performance where it struggles the most. (3) This dynamic weighting effectively reduces the overwhelming influence of head classes, since these classes tend to produce more confidently correct predictions that are down-weighted. (4) Consequently, tail classes, which inherently present more challenging classification problems, receive increased attention and a larger share of meaningful updates. (5) The intuition is that by penalizing confident yet trivial predictions less, the model has "budget" to allocate its learning capacity towards harder or rarer samples, thereby mitigating the imbalance effect. (6) Empirical results have shown that this approach can significantly boost performance on minority classes without severely degrading performance on majority classes [1]. (7) Since Focal Loss does not depend solely on class frequency, it can adapt to the evolving difficulty distribution of samples during training, making it more flexible than static frequency-based weights. (8) Though originally introduced in the context of object detection, it has been widely adopted in image classification tasks with long-tailed distributions. (9) Thus, Focal Loss stands as a prime example of a re-weighting strategy that leverages model feedback (prediction confidence) rather than static priors (class frequencies) to balance training.

Class-Balanced Loss The key insight is that each class's influence should not be linearly scaled by its frequency alone. Instead, it considers that as class size grows, its incremental value decreases, suggesting that fewer samples are needed from frequently seen classes to establish robust representations. By formulating a class-specific weight derived from the effective sample number, CB Loss prevents large classes from dominating the training gradient simply due to their sheer volume of examples. This leads to a more stable and theoretically grounded re-weighting mechanism than naive inverses of class frequency, and can yield better generalization on minority classes. The concept of effective number is computed using a parameter γ to determine how quickly the marginal benefit of additional samples decreases. This approach balances the training signal by giving classes a weight proportional to the inverse of their effective number, essentially normalizing the influence of different classes to a more comparable scale [1]. As a result, CB Loss ensures that each class, regardless of its raw frequency, contributes meaningful gradients that reflect both its representation level and its effective complexity. This method elegantly merges the ideas of frequency-based weighting with a diminishing return model, acknowledging that overly abundant classes do not linearly improve overall performance. With this refined weighting scheme, CB Loss tends to improve recognition of rare categories without unduly harming head class accuracy.

Balanced Softmax Loss Balanced softmax loss applies a correction to the logits rather than just weighting the final loss terms can more directly counter the skewed gradient flow that results from class imbalance [41]. By multiplying logits by the class frequencies, it effectively adjusts the decision boundaries, ensuring that classes with fewer samples are not overshadowed by the dense representation of majority classes. This adjustment encourages the model to produce a more uniform posterior distribution that accurately reflects true class priors, both during training and inference. Consequently, the network learns a more balanced internal representation, improving recognition of tail classes without overly penalizing head classes. It can be seen as a more direct approach to re-weighting at the logit level, contrasting with methods that focus on re-weighting the loss function after probability computation. Empirical results show that BS Loss can be more effective than other re-weighting methods, as it corrects the bias at its source rather than compensating after the fact.
The implementation details for the balanced softmax loss can be viewed in algorithm 1 in appendix D.

Equalization Loss (1) In highly imbalanced scenarios, each positive sample from a head class can produce negative gradients for all other classes, including tail classes. Over time, this can discourage the network from correctly identifying these minority classes. (2) EQ Loss counteracts this by selectively down-weighting the negative gradients that arise when tail classes appear as negative samples for majority classes [1]. (3) By reducing the negative influence on tail classes, EQ Loss ensures that the model does not develop an overly "head-biased" representation that fails to recognize minority categories. (4) This strategy highlights a subtlety in the re-weighting paradigm: it is not only about boosting tail class importance, but also about preventing undue suppression of these classes through negative gradients. (5) However, excessively down-weighting negative gradients may impair the model's discriminative power. To address this, adaptive variants, such as ACSL and Equalization v2, adjust the amount of gradient suppression based on the model's prediction confidence [42, 1]. (6) This adaptive mechanism allows the network to selectively protect tail classes when needed, while still maintaining a strong discriminative boundary between classes. (7) The result is a more balanced training signal that prevents rare classes from being "washed out," thereby improving long-tailed recognition. (8) EQ Loss exemplifies a more fine-grained approach to re-weighting, intervening at the gradient level to ensure fair treatment of minority categories. (9) By focusing on the gradient flow rather than just the static weighting of loss values, EQ Loss and its successors illustrate the evolving sophistication of class-sensitive re-weighting techniques.

LDAM Loss (1) Margin-based losses generally improve generalization by enforcing a minimum separation between class decision boundaries. (2) LDAM incorporates class-dependent margins that are inversely related to class frequencies, providing larger margins to tail classes and smaller margins to head classes [1]. (3) By increasing the margin for minority classes, it effectively lowers their decision boundary threshold, making it easier for the model to confidently classify these classes and reducing their misclassification rates. (4) This approach contrasts with re-weighting methods: while re-weighting modifies the magnitude of the loss contributions, re-margining directly influences how decision boundaries are formed in the embedding space. (5) The motivation behind LDAM is that the imbalance problem is not solely a matter of loss amplitude, but also how easily minority classes can carve out their own regions in the feature space. (6) With larger margins for tail classes, LDAM encourages more robust and discrimination-friendly representations for these classes, thereby counteracting the bias induced by uneven sample distributions. (7) As a result, the final classifier tends to perform better on rare categories, as it learns to keep them well-separated from other classes, even when their sample counts are low. (8) Combining LDAM with re-weighting or re-sampling techniques can yield even stronger improvements, as each method addresses a complementary aspect of the imbalance issue. (9) LDAM underscores that balancing the training process can be achieved not only by adjusting loss weights, but also by reshaping the geometry of the decision boundaries in class-sensitive ways.

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.

Dataset Specifications The experiments in this thesis are conducted using the CIFAR-100 [21] benchmark. with imbalance factor 0.01. The CIFAR-100 is split into a training set and a test set. This creates a training set of 45,000 samples, and a test set of 5,000 samples. Furhter, the CIFAR-100-LT was created from the training set with an imbalance factor of 0.01 with the number of samples per class ranging from 450 in the largest class to just 4 in the smallest. The test set mirrors this distribution and maintains the same imbalance factor.

To further analyze model performance across different class frequencies, the imbalanced test set is divided into three subsets: the head test set, containing the top one-third most frequent classes, the middle test set, which includes the middle one-third of classes, and the tail test set. The validation set is the original CIFAR-100 test set with 10,000 samples distributed equally across the 100 classes.

Training Configuration All experiments are implemented using the PyTorch deep learning framework (Paszke et al., 2019), with models initialized using pretrained weights on the ImageNet-1K dataset to leverage transfer learning. Four model architectures are used: MobileNetV2 (Sandler et al., 2018), ResNet50V2 (He et al., 2016), ConvNeXt Base (Liu et al., 2022), and ViT-B/16 (Dosovitskiy et al., 2021). Each model is modified to adapt to the CIFAR-100 dataset by replacing the final classification layer with a 100-class fully connected layer.

Training is performed over 90 epochs with a batch size of 128. The Adam optimizer (Kingma and Ba, 2014) is employed with an initial learning rate of 0.001, which is reduced by a factor of 0.1 every 30 epochs using a StepLR scheduler. The training is conducted on a high-performance computing system equipped with four NVIDIA TITAN X GPUs, utilizing PyTorch's DataParallel module to support multi-GPU training.

Evaluation Metrics To evaluate model performance, the primary metric used is the top-1 accuracy, which measures the proportion of correctly classified samples. Additionally, the F1 score is reported to capture the balance between precision and recall, with the macro F1 score being used for balanced datasets such as the validation set and balanced test set, and the weighted F1 score for imbalanced datasets. The head, middle, and tail subsets of the imbalanced test set are evaluated separately to analyze performance across class frequencies. These metrics provide insight into how well the models perform on classes with varying sample distributions.

Hardware and Software The experiments are conducted on a high-performance system running Ubuntu 22.04, equipped with four NVIDIA TITAN X GPUs (12GB each) and 125GB of RAM. The software environment includes Python 3.11.8, PyTorch 1.7 or higher, and Torchvision 0.8.0 or higher. CUDA version 12.4 is used to optimize GPU computations.

Reproducibility To ensure reproducibility, a fixed random seed of 42 is used across all experiments. This seed controls randomness in Python, NumPy, and PyTorch, with GPU computations enforced to be deterministic by setting the cudnn.deterministic flag to True. Configuration files in YAML format are used to document hyperparameters, dataset settings, and model configurations, allowing for exact replication of experiments. Datasets and model checkpoints are saved to facilitate reuse and further analysis.

Chapter 5

Results and Discussion

This chapter presents the experimental results, comparing model performances across head, middle, and tail classes, and discusses the impact of different model architectures combined with different loss functions. First, the main findings are presented, followed by a detailed analysis of the results across experiments, and lastly a summary and 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 ResNet50 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 ResNet50 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 ResNet50 (Acc1: 0.7916). See Tables 5.4 and 5.8 for reference. These findings, however, require further statistical analysis to confirm their significance and that the observed differences are not simply due to random variability.

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 trained with balanced CIFAR-100 (Acc1: 93.95 %) among all model architectures investigated in this thesis [47]. This discrepancy suggests that the ViT-B/16 configuration used for the experiments in this thesis may not be well-suited for smaller-scale datasets such as CIFAR-100 without careful optimization as discussed in sections 2.2.3 and 3.3.1 TODO: write these sections.

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 each loss design on a given model, but not directly comparing loss designs or models. It provides an overview of all findings to understand how each combination of architecture and loss function behaves.

TODO: Include that all models are trained with a balanced CIFAR-100 to provide a baseline for the class re-balancing strategies.

5.2.1 MobileNetV2

Results from Balanced Training

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**

From Table 5.1, Balanced Softmax Loss achieves the best accuracy on the balanced test dataset (Acc1: 0.8034) and the highest accuracy on tail classes (Acc1: 0.9211). Not surprisingly, Softmax, Weighted Softmax, and Class-Balanced Loss yield the same accuracies across all test datasets due to shared cross-entropy-based designs that become indistinguishable when trained on balanced data according to equations (2.6), (2.7), and (2.10) described in section 2.3.1. Overall, all methods achieve comparable results to the softmax baseline, while outperforming the best published result for MobileNetV2 trained with CIFAR-100, as seen in table 5.9.

TODO: This paragraph could potentially be moved to an overall discussion section, as it includes a discussion of other model behavior. Although LDAM Loss demonstrates lower overall accuracy, its high tail-class accuracy matches that of Balanced Softmax and Equalization Loss, suggesting that tail-class performance may have reached a saturation point due to the limited number of samples in tail classes. This notion is further supported by identical performance across multiple long-tailed subsets with other backbones (see tables 5.3 and 5.7), suggesting potential limitations in class-specific data quality or quantity. These limitations could be investigated through testing on a larger dataset.

Results from Long-Tailed Training

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

From Table 5.2, Balanced Softmax Loss outperforms other losses in terms of balanced test accuracy (0.5796) and especially tail classes (0.4211). Although Softmax and Focal Loss surpass Balanced Softmax on the long-tailed dataset and head classes, the improvements Balanced Softmax provides for tail and middle classes are notable. Class-Balanced Loss shows an overwhelming underperformance compared with the balanced training in table 5.1, and now appears severely limited, possibly due to implementation issues that must be investigated TODO: investigate. The difference in performance patterns compared to the softmax baseline shows that loss functions designed for imbalance can provide more nuanced performance gains.

5.2.2 ResNet50

Results from Balanced Training

Table 5.3: Evaluation results for ResNet50 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, Softmax, Weighted Softmax, and Class-Balanced Loss yield identical performance when trained on balanced data due to their cross-entropy term in equations (2.10), (2.7), and (2.10). Balanced Softmax Loss demonstrates strong performance on the long-tailed test set (0.8430) and head classes (0.8460), while still performing comparably on tail classes. Overall, the performance of the class-rebalancing methods compare to that of the softmax baseline. The best performing design yield an accuracy of 83.24%, not far behind the best published result for ResNet50 with an accuracy of 86.90% as seen in table 5.9.

Results from Long-Tailed Training

Table 5.4: Evaluation results for ResNet50 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, Balanced Softmax Loss significantly improves tail-class accuracy (0.6053), outperforming other losses, while also outperforming on balanced and middle sets. Softmax Loss achieves the best performance on the long-tailed dataset and head classes, indicating that while Balanced Softmax improves minority-class accuracy, Softmax remains strong for majority classes. Class-Balanced Loss and LDAM Loss both show underwhelming results, pointing toward the necessity of further investigation for Class-Balanced Loss TODO: investigate or combining LDAM with DRW [20].

5.2.3 ViT-B/16

Results from Balanced Training

From Table 5.5, LDAM Loss achieves the overall best performance for ViT-B/16 trained on balanced data. However, the overall performance remain far below the benchmark results for Vision Transformers on CIFAR-100, see table 5.9. This suggests that further adjustments are needed, possibly including extended training time, data augmentation, or different optimizers, as ViT architecture differs from CNNs, as describe in section 2.2.3.

Results from Long-Tailed Training

From Table 5.6, performance remains low across all loss functions. Balanced Softmax Loss slightly improves results on balanced and middle categories, while

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**

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

Softmax Loss achieves the highest accuracy on the long-tailed dataset and head classes.The best accuracy on tail classes (0.1053) is achieved by Class-Balanced Loss, though it fails everywhere else. This pattern suggests that the ViT-B/16 architecture may not be sufficient for robust performance on smaller-scale, long-tailed datasets and may require major adjustments in the training pipeline, see section 2.2.3.

5.2.4 ConvNeXt Base

Results from balanced Training Dataset

From Table 5.7, Balanced Softmax Loss achieves the best balanced test accuracy (0.8364) while the Softmax architechtures tie for top performance on the long-tailed dataset, head classes, and tail classes. The saturation of tail-class performance is likely due to the quality or quantity of tail classes. Balanced Softmax's advantage on balanced data may stem from its logit-adjustment mechanism, which can enhance generalization even when classes are balanced [41] TODO: investigate. Although the best published result for a ConvNext architecture trained with CIFAR-100 is the Conv2NeXt with an accuarcy of 83.82% (see table 5.9), the best perfomance of on the balanced test set is in alignment, yielding an accuracy of 83.64%.

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

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, Balanced Softmax Loss outperforms on the balanced test set (0.6460) and tail classes (0.5789) by a considerable margin. Softmax achieves the best performance on the long-tailed dataset and head classes, while Equalization Loss leads on middle classes. The poor performance of Class-Balanced Loss stands out, calling for further investigation into its implementation TODO: investigate.

5.3 Benchmarks

Table 5.9 compares the best results from the models trained on the balanced CIFAR-100 dataset in this thesis with published benchmarks using the same or similar architectures. This comparison serves as a reference to ensure the results align with expectations and to identify any potential discrepancies.

Notably, there are no published benchmarks for ConvNeXt Base trained on CIFAR-100. The closest benchmark available is from Feng et al. [48], where Conv2NeXt achieved a top-1 accuracy of 83.82%, which is highly similar to the result obtained here.

For MobileNetV2, the results in this thesis surpass the published benchmarks. This could be attributed to improved loss function designs. On the other hand, ResNet50 exhibits a slightly lower accuracy compared to benchmarks, which may be explained by differences in optimization strategies, such as the use of Adam optimizer instead of SGD [45].

The most significant deviation is observed with ViT-B/16, where the accuracy falls far short of the benchmark. This discrepancy suggests that the configuration for ViT-B/16 may not optimized for training with small-scale datasets, see section 2.2.3.

Overall, these results indicate that the experimental setup is effective for most models but highlights potential limitations for transformer-based architectures such as ViT-B/16. Future investigation is necessary to understand and address these discrepancies.

Table 5.9: Comparison of model performance on balanced CIFAR-100 with published benchmarks.

Model	**Result (Acc1)**	**Benchmark Result (Acc1)**
MobileNetV2	80.34%	73.20% [49]
ResNet50	83.24%	86.90% [50]
ViT-B/16	59.06%	93.95% [47]
ConvNeXt Base	83.64%	83.82% [48]

5.4 Performance Comparisons

The models are compared by taking the mean of the results of the loss functions for each model. Excluding the generally underperforming Class-Balanced Loss, the mean and standard deviation across all loss designs for each model is plotted in figure 5.1. TODO: Include a table of the values. Here, the performance of the models MobileNetV2, ResNet50, ViT-B/16, and ConvNeXt Base is shown across the evaluation categories: balanced, long-tailed, head, middle, and tail. Each model is slightly offset along the x-axis to avoid overlap. Each point represents the mean accuracy of a model for a specific category and the bars represent the standard deviation. Again, the mean and standard deviation excludes those of the Class-Balanced Loss. The error bars indicate the variability in accuracy across different loss functions for each model and category. A longer error bar demonstrate a less consistent performance, dependent on the chosen loss function, while shorter error bars indicate consistant performance adhering to the specific model.

From figure 5.1, the shorter errorbars for MobileNetV2 indicate that its architecture may be more robust to the type of re-weighting strategy used than other architechtures. Contrary, the ResNet50V2 performance on tail classes shows greater sensitivity to the chosen loss design. ViT-B/16 shows a consistently lower mean accuracy, indicating that transformer-based approaches may require additional adaptations for smaller-scale or imbalanced datasets, as discussed in 3.3.1. ConvNeXt Base maintains strong and stable performance across several categories, benefiting from Balanced Softmax Loss as seen in table 5.8.

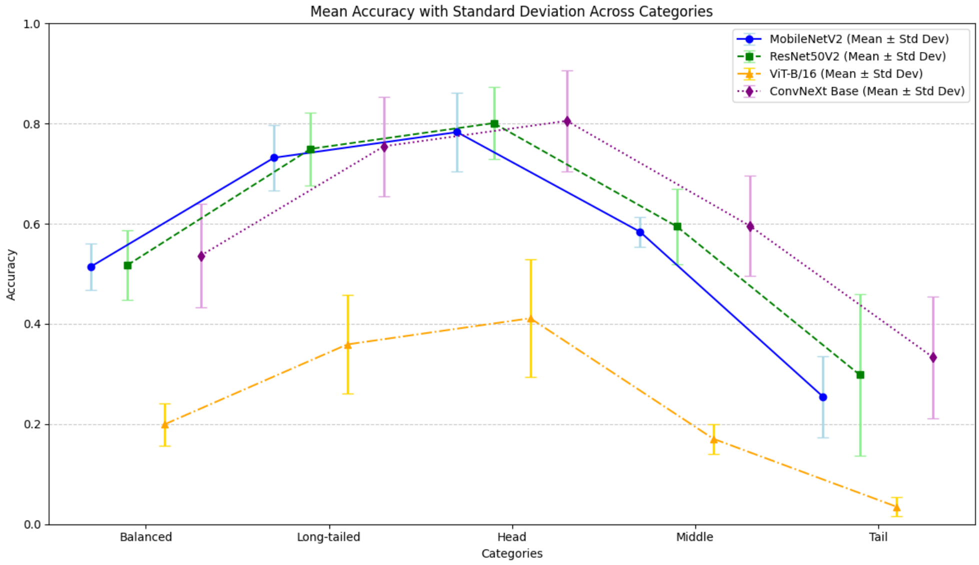

Figure 5.1: Mean Accuracy with Standard Deviation Across Categories for MobileNetV2, ResNet50, ViT-B/16, and ConvNeXt Base trained with CIFAR-100-LT. These values are excluding those of the Class-Balanced Loss.

The Balanced Softmax Loss emerges as the overall best performing loss design across all models with the least variance in performance across test categories, as figures 5.2, 5.3, 5.5, and 5.4 illustrate.

TODO: Include variance across test categories in either figures or tables.

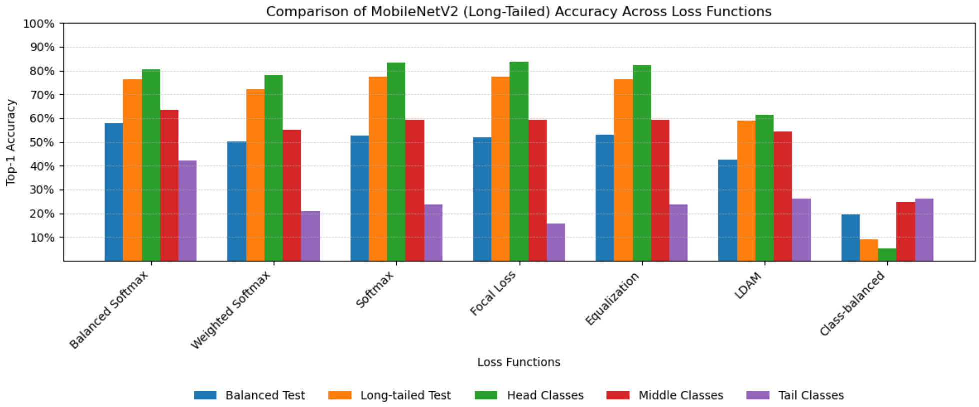

Figure 5.2: MobileNetV2 top 1 accuracy across test categories (Balanced, Long-tailed, Head, Middle, Tail) for different loss functions.

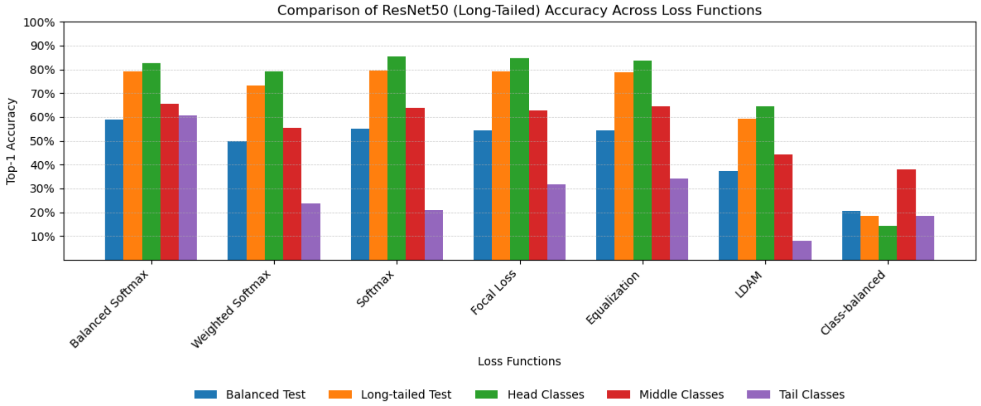

Figure 5.3: ResNet50 top 1 accuracy across test categories (Balanced, Long-tailed, Head, Middle, Tail) for different loss functions.

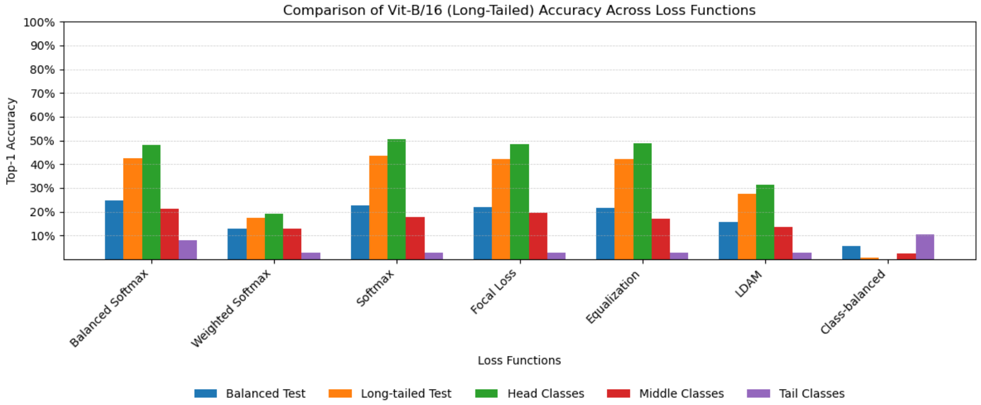

Figure 5.4: ViT-B/16 top 1 accuracy across test categories (Balanced, Long-tailed, Head, Middle, Tail) for different loss functions.

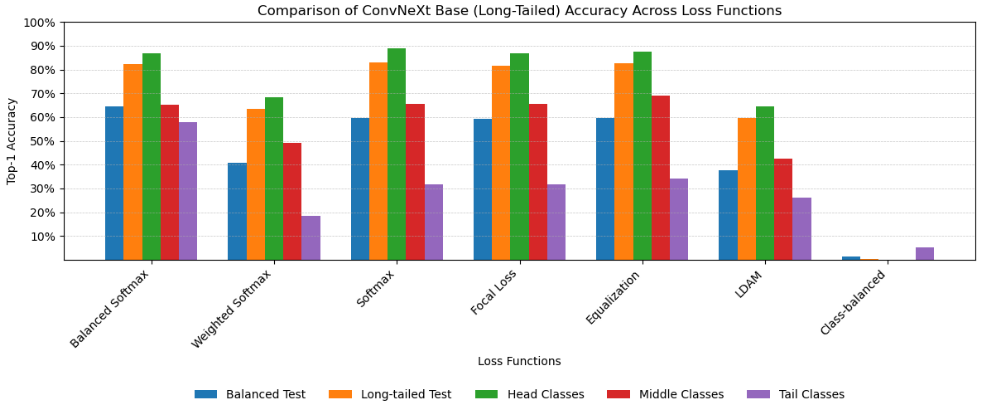

Figure 5.5: ConvNeXt Base top 1 accuracy across test categories (Balanced, Long-tailed, Head, Middle, Tail) for different loss functions.

This result is further underlined from figure 5.6, where the mean and standard deviation of the loss functions across ResNet50, MobileNetV2, ViT-B/16, and ConvNeXt is calculated and plotted. Here, the Balanced Softmax Loss shows a better average perfomance on tail classes, while also displaying the smoothest variation across evaluation categories TODO: calculate variance, ignoring the Class-Balanced loss. However, the standard deviation suggests that the perfomance of the Balanced Softmax Loss on tail classes depend on the model architecture, as earlier discussed. Table 5.10 display the exact values of the mean and standard deviations shown in figure 5.6.

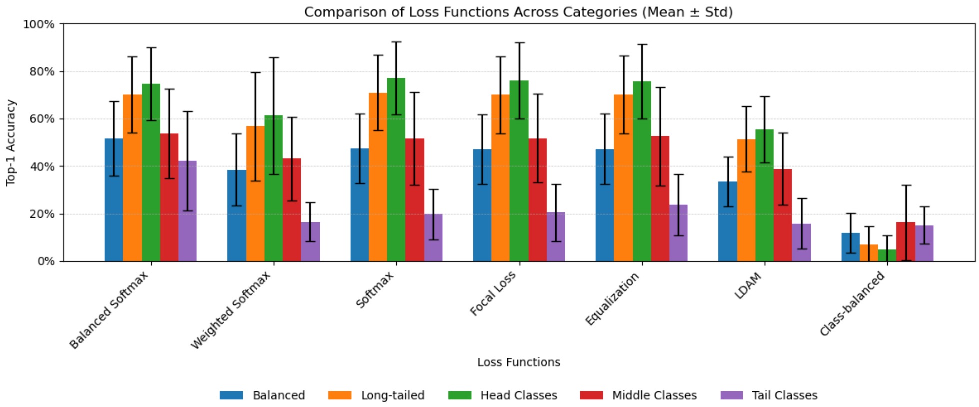

Figure 5.6: Performance trends of different loss functions across evaluation categories. Error bars indicate the standard deviation of accuracy across models, highlighting variability in performance for each loss function.

Table 5.10: Performance Summary Across Categories (Mean ± Std)

Loss Function	Balanced	Long-tailed	Head	Middle	Tail
Balanced Softmax	0.5156 ± 0.1577	0.7010 ± 0.1610	0.7462 ± 0.1540	0.5385 ± 0.1881	0.4211 ± 0.2097
Weighted Softmax	0.3841 ± 0.1522	0.5671 ± 0.2290	0.6123 ± 0.2462	0.4320 ± 0.1761	0.1645 ± 0.0819
Softmax	0.4758 ± 0.1466	0.7093 ± 0.1587	0.7710 ± 0.1537	0.5163 ± 0.1970	0.1974 ± 0.1061
Focal Loss	0.4701 ± 0.1462	0.7008 ± 0.1624	0.7598 ± 0.1599	0.5178 ± 0.1876	0.2039 ± 0.1211
Equalization	0.4721 ± 0.1494	0.7010 ± 0.1629	0.7571 ± 0.1558	0.5252 ± 0.2072	0.2368 ± 0.1289
LDAM	0.3504 ± 0.1224	0.5187 ± 0.1443	0.5548 ± 0.1792	0.3947 ± 0.1886	0.1587 ± 0.1078
Class-balanced	0.1172 ± 0.0814	0.0711 ± 0.0841	0.0490 ± 0.0624	0.1628 ± 0.1477	0.1513 ± 0.0946

TODO: Comparison of overall model performance. Potentially statistical analysis, e.g. ANOVA, Tukey's HSD.

5.5 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.
- Argue why the test data for CIFAR-100 was created from the training data instead of the validation data, and why or why not it would have been better to split the validation data instead.

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 ResNet50 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.

TODO: Influence of optimizer `https://arxiv.org/pdf/2007.07314v2` `https://openreview.net/pdf?id=rk6qdGgCZ`

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

The goal of this thesis was to investigate the impact of loss functions combined with different backbones trained with long-tailed datasets in order to see the impact of the different methods on tail-class accuracy.

6.2 Future Work

Conduct experiments on ImageNet or other larger datasets. Train ViT-B/16 with larger dataset and change some of the parameters.
Train CIFAR-100-LT with LDAM-DRW and different loss functions.

Bibliography

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

[2] Chongsheng Zhang et al. *A Systematic Review on Long-Tailed Learning.* 2024. arXiv: 2408.00483 [cs.LG]. URL: https://arxiv.org/abs/2408.00483.

[3] N. V. Chawla et al. "SMOTE: Synthetic Minority Over-sampling Technique". In: *Journal of Artificial Intelligence Research* 16 (June 2002), pp. 321–357. ISSN: 1076-9757. DOI: 10.1613/jair.953. URL: http://dx.doi.org/10.1613/jair.953.

[4] Hui Han, Wen-Yuan Wang, and Bing-Huan Mao. "Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning". In: *Proceedings of the 2005 International Conference on Advances in Intelligent Computing - Volume Part I.* ICIC'05. Hefei, China: Springer-Verlag, 2005, pp. 878–887. ISBN: 3540282262. DOI: 10.1007/11538059_91. URL: https://doi.org/10.1007/11538059_91.

[5] Xi Yin et al. *Feature Transfer Learning for Deep Face Recognition with Under-Represented Data.* 2019. arXiv: 1803.09014 [cs.CV]. URL: https://arxiv.org/abs/1803.09014.

[6] 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.

[7] Luis Perez and Jason Wang. *The Effectiveness of Data Augmentation in Image Classification using Deep Learning.* 2017. arXiv: 1712.04621 [cs.CV]. URL: https://arxiv.org/abs/1712.04621.

[8] Connor Shorten and Taghi M. Khoshgoftaar. "A survey on image data augmentation for deep learning". In: *Journal of Big Data* 6.1 (2019), pp. 1–48.

[9] Hongyi Zhang et al. *mixup: Beyond Empirical Risk Minimization.* 2018. arXiv: 1710.09412 [cs.LG]. URL: https://arxiv.org/abs/1710.09412.

[10] Sangdoo Yun et al. *CutMix: Regularization Strategy to Train Strong Classifiers with Localizable Features.* 2019. arXiv: 1905.04899 [cs.CV]. URL: https://arxiv.org/abs/1905.04899.

[11] Zhengzhuo Xu, Zenghao Chai, and Chun Yuan. *Towards Calibrated Model for Long-Tailed Visual Recognition from Prior Perspective.* 2021. arXiv: 2111.03874 [cs.CV]. URL: https://arxiv.org/abs/2111.03874.

[12] Jianfeng Wang et al. *RSG: A Simple but Effective Module for Learning Imbalanced Datasets.* 2021. arXiv: 2106.09859 [cs.CV]. URL: https://arxiv.org/abs/2106.09859.

[13] Bingyi Kang et al. *Decoupling Representation and Classifier for Long-Tailed Recognition*. 2020. arXiv: `1910.09217 [cs.CV]`. URL: `https://arxiv.org/abs/1910.09217`.

[14] Boyan Zhou et al. *BBN: Bilateral-Branch Network with Cumulative Learning for Long-Tailed Visual Recognition*. 2020. arXiv: `1912.02413 [cs.CV]`. URL: `https://arxiv.org/abs/1912.02413`.

[15] Xudong Wang et al. *Long-tailed Recognition by Routing Diverse Distribution-Aware Experts*. 2022. arXiv: `2010.01809 [cs.CV]`. URL: `https://arxiv.org/abs/2010.01809`.

[16] 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`.

[17] 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`.

[18] 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`.

[19] 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`.

[20] 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`.

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

[22] 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.

[23] 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.

[24] 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`.

[25] 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`.

[26] 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.

[27] 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`.

[28] 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.

[29] 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`.

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

[31] 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`.

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

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

[34] 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`.

[35] 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`.

[36] 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`.

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

[38] 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`.

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

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

[41] 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`.

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

[43] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. `http://www.deeplearningbook.org`. MIT Press, 2016.

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

[45] Aditya Krishna Menon et al. *Long-tail learning via logit adjustment*. 2021. arXiv: `2007.07314 [cs.LG]`. URL: `https://arxiv.org/abs/2007.07314`.

[46] PyTorch Contributors. *torchvision.datasets.CIFAR100*. Accessed: 2024-12-13. 2024. URL: `https://pytorch.org/vision/0.17/generated/torchvision.datasets.CIFAR100.html`.

[47] 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`.

[48] Jianwei Feng et al. "Conv2NeXt: Reconsidering Conv NeXt Network Design for Image Recognition". In: *2022 International Conference on Computers and Artificial Intelligence Technologies (CAIT)*. 2022, pp. 53–60. DOI: `10.1109/CAIT56099.2022.10072172`.

[49] Geon Park et al. *BiTAT: Neural Network Binarization with Task-dependent Aggregated Transformation*. 2022. arXiv: `2207.01394 [cs.CV]`. URL: `https://arxiv.org/abs/2207.01394`.

[50] 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`.

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

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

[53] 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.

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.

Appendix B

Benchmark Specifications

B.1 MobileNetV2 on CIFAR100

Table B.1: Comparison of MobileNetV2 on CIFAR100 with their study (Procedure A3) [50].

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

Appendix C

Experimental Setup Details

C.1 Dataset Specifications

Table C.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 C.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 C.1.

Table C.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 C.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

C.2 Data Preprocessing

Table C.3 summarizes the preprocessing steps applied to the training, validation, and test datasets. TODO: reference CIFAR100 statistics.

Table C.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]

C.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 [51]. The head is replaced with a 100-clas fully connected layer.

ConvNeXt Base The ConvNeXt Base architecture is pretrained on the ImageNet-1K dataset [52], and the final layer is replaced with a 100-class fully connected layer.
The specifications of the model architectures can be seen in table C.4 TODO: make this table prettier.

Table C.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

C.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 C.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.

C.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.

C.6 Hardware and Software Configurations

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

C.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

C.6.2 Software

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

C.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.

C.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 [51] 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 [53].

- Class-Balanced Loss.

Appendix D

Loss Implementation

This appendix provides pseudo code for the implementation of the deep long-tailed learning strategies used in this thesis.

Algorithm 1 BalancedSoftmaxLoss

```
Require: Logits $\mathbf{Z} \in \mathbb{R}^{N \times C}$, Targets $\mathbf{y} \in \{1, \dots, C\}^N$, Class counts $\mathbf{N} = [N_1, \dots, N_C]$
Ensure: Loss value $\ell$
Compute class priors: $p_i \leftarrow \frac{N_i}{\sum_{j=1}^{C} N_j}, \forall i \in \{1, \dots, C\}$
Initialize a small constant $\epsilon \leftarrow 10^{-6}$
for $n = 1$ to $N$ do
    for $i = 1$ to $C$ do
        $\widetilde{Z}_{n,i} \leftarrow Z_{n,i} + \log(p_i + \epsilon)$
    end for
end for
Compute the balanced softmax probabilities:
    $$P_{n,i} \leftarrow \frac{\exp(\widetilde{Z}_{n,i})}{\sum_{k=1}^{C} \exp(\widetilde{Z}_{n,k})}$$
Compute cross-entropy loss:
    $$\ell \leftarrow -\frac{1}{N} \sum_{n=1}^{N} \log(P_{n,y_n})$$
return $\ell$
```