Deep Learning Techniques for Long-Tailed Image Classification: A Comparative Study of Loss Designs and Model Architectures

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, characterized by few classes with many samples followed by many classes with few samples, pose significant challenges for image classification tasks. This thesis investigates deep learning techniques tailored to address long-tailed class distributions, focusing on class-sensitive learning and model architecture selection. Four state-of-the-art models (ResNet-50, MobileNetV2, ConvNeXt-Base, and ViT-B/16) are trained on both balanced and long-tailed versions of the CIFAR-100 dataset, where the balanced training serves as a baseline for the performances on the long-tailed training. Furthermore, the study evaluates the loss functions; Softmax Cross-Entropy Loss, Weighted Softmax Cross-Entropy Loss, Focal Loss, Class-Balanced Loss, Balanced Softmax Loss, LDAM Loss, and Equalization Loss, to assess their effectiveness in improving performance on tail classes while maintaining accuracy across head and middle classes.

The goal is to provide insights into the interplay between loss design and model architecture, offering practical recommendations for handling long-tailed distributions in real-world datasets. Experiments revealed that Balanced Softmax Loss consistently surpassed other loss designs on tail classes while preserving accuracy on head classes when trained with Convolutional Neural Network (CNN) architectures with the best tail-class performance of 60.53% top-1 accuracy on ResNet-50. Additionally, the Vistion Transformer (ViT) architecture, ViT-B/16, persistently underperformed on both the balanced and long-tailed versions of CIFAR-100, not only lagging far behind the CNN-based architectures but also failing to meet the published benchmark. This highlights the need for further optimization to accomodate for small-scaled datasets, like CIFAR-100, and to unlock the full potential of the ViT models. Finally, the performance of Equalization Loss across various model architectures reveals the importance of careful selection of model and loss design, as it underperformance on all models except the more robust ConvNeXt-Base, emphasizing ConvNeXt's overall ability to maintain performance when paired with different loss designs.

These results contributes to the understanding of long-tailed learning and guides practitioners in selecting effective strategies for addressing class imbalance.

Acknowledgements

The author of this thesis would like to thank Kim Bjerge, Associate Professor at Aarhus Univeristy, for supervision, constructive feedback, and inspiring conversations throughout the development of this thesis.

The author is also deeply thankful to their family for their moral support.

Contents

Chapter 1

Introduction

Deep learning has become a prominent solution in recent years for tackling image regocnition tasks. With the availability of large datasets, i.e. ImageNet [1], along with GPUs, training of deep learning models have become easier and have led to remarkable results [2]. The trained models have shown image classification with accuracies of over 80 % [3, 4], and hence the interest in deep learning for image recognition is high. However, the high accuracies are usually achieved on models trained with balanced datasets, while most real-world datasets are skewed in samples per class [5, 6, 7].

This thesis addresses the challenge of long-tailed datasets in image classification, where most classes contain only a few samples. A long-tailed class distribution causes the model to primarily learn features from majority (head) classes, leading to a bias toward these dominant classes, and, as a result, the trained model struggles to recognize inputs from minority (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 [8, 9]. The aim of this thesis is to examine class re-balancing techniques to tackle the long-tailed problem in deep learning.

1.1 Problem Definition

This thesis explores deep learning techniques for addressing long-tailed distributions in image classification, focusing on the interplay between backbone architectures and class-sensitive learning methods.

Four deep learning state-of-the art models are investigated, including three Convolutional Neural Networks (CNNs), and one Vision Transformer (ViT). The CNNs investigated are ResNet-50 [3], MobileNetV2 [10], and ConvNeXt-Base [11], while the ViT investigated is the ViT-B/16 [4]. These particular models were chosen because of their individual strenghts. ResNet-50 is a variant of the ResNet architecture [3], offering a strong baseline, thus providing a standard for comparison. MobileNetV2, based on the ResNet architecture, respresents models designed for environments with limited resources. ConvNeXt-Base is a newer interpretation of CNNS, incorporating design principles from ViTs, bridging the gap between tradional CNNs and ViTs, while ViT-B/16 represents traditional ViTs. All four

models leverages transfer learning, pretrained on ImageNet [1], and subsequently trained on a balanced version of the CIFAR-100 [12] dataset, which serves as a baseline for comparing the training on the long-tailed version of CIFAR-100 (CIFAR-100-LT) [13].

The long-tailed techniques investigated in this thesis focus on class re-balancing, with an emphasis on class-sensitive learning through loss functions. All models are trained using Softmax Cross-Entropy (CE) loss [14], Weighted Softmax Cross-Entropy (WCE) loss [9], Focal Loss (FL) [15], Class-Balanced (CB) loss [16], Balanced Softmax (BS) loss [17], Label-Distribution-Aware-Margin (LDAM) loss [13], and Softmax Equalization Loss (SEQL) [18]. By including these loss designs, this thesis investigates the effectiveness of each in mitigating the challenges posed by class imbalance.

The selection of these loss functions is based on their alignment with class-sensitive approaches highlighted in the paper *Deep Long-Tailed Learning: A Survey* by Zhang et al. [9]. Starting with the standard CE loss as a baseline, the study progresses through increasingly sophisticated loss functions designed to improve model performance on underrepresented tail classes while maintaining accuracy across the entire dataset. This comparative analysis aims to provide insights into how specific loss functions interact with model architectures in addressing the challenges of long-tailed distributions.

By evaluating the effectiveness of different models and techniques, this study seeks to guide practitioners in selecting appropriate strategies to address class imbalance, improve model performance on underrepresented classes, and achieve more balanced predictions.

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.

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 choice of model architecture and the degree of imbalance in the dataset.

1.1.3 Approach

This master's thesis consists of five stages, described below:

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

Stage two is to generate a long-tailed version of CIFAR-100 that can be used for training and comparisons of methods. This is inspired by Cao et al. [13], who uses the CIFAR-100-LT dataset for experiments.

Stage three is the selection of model architectures and class-sensitive methods.

Stage four is training and evaluation of the models with different loss design on both the balanced and long-tailed versions of CIFAR-100.

Stage five is a comparison of the methods.

1.1.4 Scope of this thesis

This thesis seeks to apply deep long-tailed learning methods to image classification tasks with an emphasis on loss design to mitigate the effect imposed by class imbalance. The loss designs include CE loss, FL, WCE loss, CB loss, BS loss, SEQL, and LDAM Loss. The experiments are conducted on four backbone architectures: ResNet-50, MobileNetV2, ConvNeXt-Base, and ViT-B/16. These are trained on the CIFAR-100 dataset with both a balanced and synthetically generated long-tailed version. The evaluation metric is the top-1 accuracy, with attention paid to performance on head, middle, and tail classes. This thesis does not explore other long-tailed learning approaches such as re-sampling, information augmentation, or model architecture modifications.

1.2 Related Work

The challenge of long-tailed datasets has been extensively studied in the literature, and to overcome the challenge of long-tailed data, various methods has been proposed [9, 19]. Zhang et al. [9] conducted a comprehensive survey on long-tailed learning techniques, categorizing them into three main strategies: class re-balancing, information augmentation, and module improvement. These are further divided into sub-categories: re-sampling, cost-sensitive learning, logit adjustment, transfer learning, data augmentation, representation learning, classifier design, decoupled training, and ensemble learning, as seen in Figure 1.1.

This section highlights methods relevant to the experiments conducted in this thesis; for additional details on other techniques, refer to the references cited in Zhang et al.'s survey.

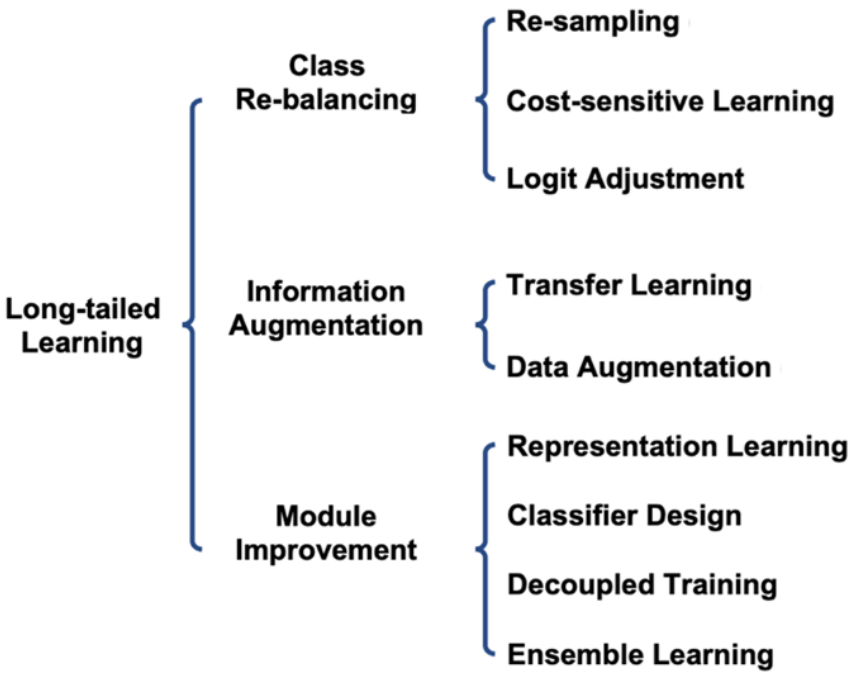


Figure 1.1: Proposed categorization of long-tailed methods by *Zhang et al.* [9]. There are three main categories: class re-balancing, information augmentation, and module improvement. These are further divided into sub-categories: re-sampling, cost-sensitive learning, logit adjustment, transfer learning, data augmentation, representation learning, classifier design, decoupled training, and ensemble learning.

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) [20]. ROS involves duplicating random samples from the minority classes, whereas RUS involves removing samples from the majority classes [9]. 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 [9]. Another re-sampling technique, called Synthetic Minority Over-Sampling Technique (SMOTE) [20], involves synthetical generation of instances of the minority classes based on the existing data.

Re-weighting Re-weighting techniques attemps to address class imbalance by assigning weights to classes or samples during training. The vanilla approach involves weighting classes proportional to the inverse sample frequency [9]. Cui et al. [16] propose re-weighting by the inverse effective number of samples. Lin et al. [15] introduces a mechanism to down-weight well-classified examples, emphasizing harder examples during training. Tan et al. [18] proposed down-weighting tail-class loss when they serve as negative vaules for head-classes. These re-weighting strategies form the foundation for comparison in this thesis.

Margin Margin-based approaches aim to improve classification performance by increasing the separation between classes. Techniques such as Large-Margin Softmax [21] and Additive Margin Softmax [22] reduce intra-class variation and enhance inter-class separability.

LDAM loss [13] focuses on integrating margin-based strategies with class-sensitive learning to explicitly encourage larger margins for tail classes. The performance of this re-margining strategy is evaluated in this thesis.

Logit Adjustment Logit adjustment methods seeks to modify the prediction logits of deep models, either applied post-hoc to a trained model, or enforced during training [23]. BS loss [17] adjusts logits by incorporating class priors into the softmax function, ensuring a more stable probability assignment across classes. The performane of the BS loss is evaluated in this thesis.

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 [24] and transferring semantic deep features [7].

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 [25, 26]. Common augmentation methods include Mixup [27], which combines pairs of samples to create interpolated examples, CutMix [28], which replaces regions of one image with patches from another, and UniMix [29], which focuses on calibrating the feature space for long-tailed distributions. Rare-class Sample Generator (RSG) [30] focuses on enhancing tail-class performance by transferring knowledge from head classes.

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 [31, 32].

1.3 Reading Guide

The reading recommendation is to read this thesis in a linear flow, beginning with Chapter 1 followed by Chapter 2 etc. However, if the reader is familiar with the subjects of this thesis, Chapter 2 can be skipped, and the reader can go to Chaper 3. Alternatively, if the reader is familar with long-tailed methods and looking for a quick guidance, jumping directly to Chapter 5 will suffice, although it is highly recommended to read Chapter 4 for a better understanding of Chapter 5. A description of the content of the chapters is presented in the following:

Chapter 2 provides the theoretical background for the long-tailed learning methodologies relevant to the work in this thesis.

Chapter 3 describes the methodology used in this thesis, including dataset preparation, selection of model and loss designs and implementation details.

Chapter 4 present the experimental results and a discussion thereof.

Chapter 5 concludes on this thesis, reflects the goals presented in Chapter 1, and discusses future work.

Chapter 2

Background

This chapter presents the different background topics of the thesis work, which includes the long-tailed datasets, model architectures Convolutional Neural Networks (CNN) and Vision Transformers (ViT), and class-sensitive deep long-tailed learning methods.

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.

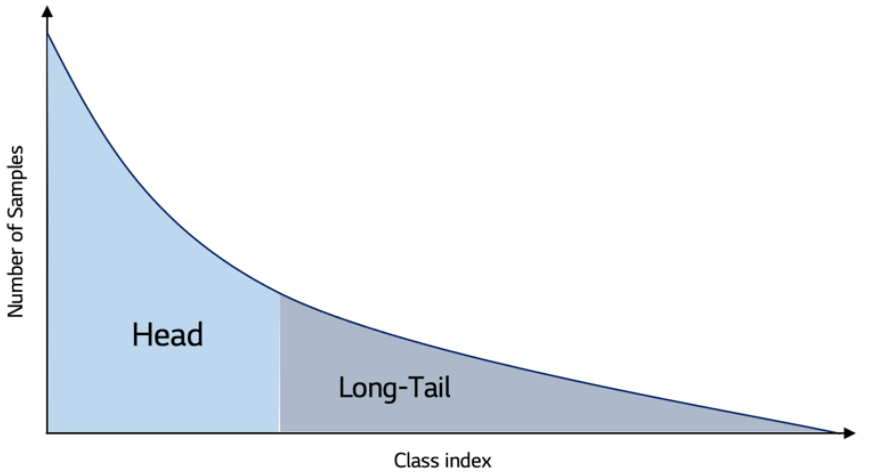


Figure 2.1: Illustration of a long-tailed distribution [33].

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 under-represented (tail classes) as depicted in Figure 2.1. Mathematically, a long-tailed training set is denoted as $X = \{x_i, y_i\}_{i=1}^{n}$, where each sample x_i has a corresponding class label y_i [9]. The total number of training samples over K classes is $n = \sum_{k=1}^{K} n_k$, where n_k represents the number of samples in class k. Let π denote the vector of label frequencies, where $\pi_k = \frac{n_k}{n}$ indicates the label frequency of class k. It can be assumed that the classes are sorted by cardinality in decreasing order. This class distribution is common for real-world datasets [34, 7], for example, the iNaturalist, a popular benchmark for image classification, exhibits a long-tailed distribution of species [8]. Other benchmarks are constructed by sampling from datasets such as ImageNet [1] by using a Pareto distribution, which simulates long-tailed class distributions with a power-law decay [9, 35, 13].

CIFAR-100-LT [13], derived from the CIFAR-100 dataset [12], 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 32×32 color images divided into 100 classes, each with 600 samples. These are further split into 500 training images and 100 testing images per class. CIFAR-100-LT is created by reducing the number of samples in certain classes of CIFAR-100 following an exponential decay en sample sizes, given by:

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

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 K is the total number of classes [13].

Other long-tailed datasets follow a Pareto distribution with number of samplers per class as followed [7]:

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

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

Class imbalance has a profound impact on model performance compared to evenly distributed datasets [5, 16]. Deep networks trained on long-tailed datasets often exhibit biased performance, favoring head classes while performing poorly on tail classes [9]. Zhang et al. [9] provide a comprehensive survey of methods addressing this challenge, categorizing current approaches into three main groups: class re-balancing, information augmentation, and module improvement. This thesis focus on class re-balancing methods which will be further explored in section 2.3.

2.2 Model Architectures

Deep learning has revolutionized image classification by introducing models capable of learning complex patterns and representations from data. Among these, CNNs and 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 [10], ResNet-50 [3], and ConvNeXt-Base [36] as the CNN architectures, and ViT-B/16 [4] as the ViT architecture. These models were chosen to represent a spectrum of design paradigms; from lightweight CNNs to transformers. By evaluating models of varying complexity, this study aims to identify architectures best suited for long-tailed learning under different resource constraints and data distributions.

2.2.1 Introduction to Deep Neural Networks

Before the introduction of CNNs and, more recently, 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, as illustrated in Figure 2.2. 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 [37, 38].

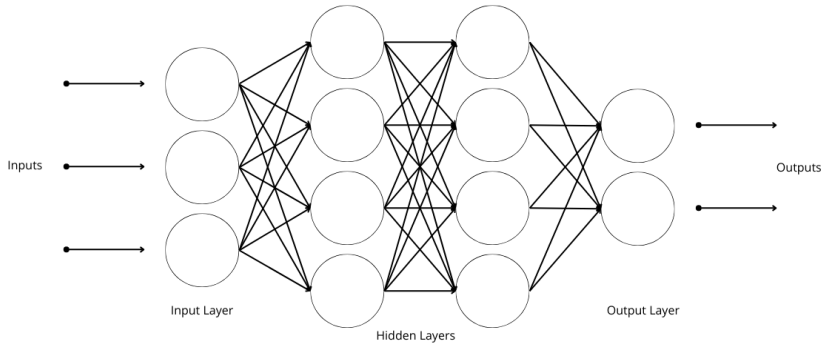


Figure 2.2: Layers of a neural network. This neural network consists of three input neurons, two hidden layers with four neurons each, and two output neurons. The layers are fully connected.

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, width, and weight 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.

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 CNNs, which introduced convolutional and pooling layers to effectively preserve and utilize the spatial information of pixels in two-dimensional images [37].

2.2.2 Convolutional Neural Networks

CNNs [39] 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 [40, 41, 37].

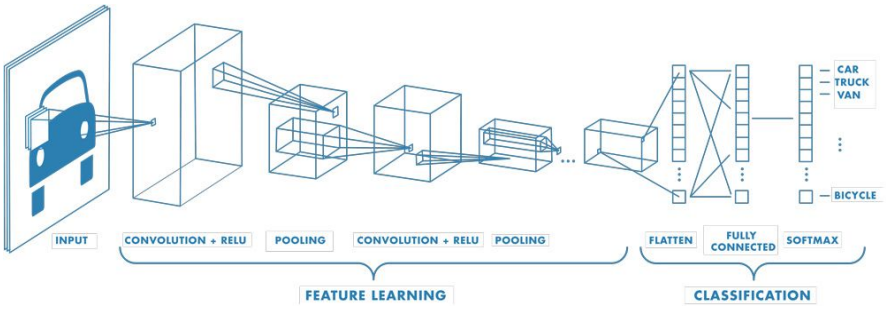


Figure 2.3: Illustration of a convolutional neural network [42].

CNNs consist of several core components, as illustrated in Figure 2.3, and have three main types of layers: convolutional layers, pooling layers, and a Fully Connected (FC) layer [14]. The convolutional layer is the first layer, and serves to extract local features by applying filters to small regions of an image. Pooling layers reduce the spatial dimensions of feature maps, providing invariance to small translations. CNNs are typically made of multiple convolution and pooling layers, but the final layer is the FC layer. Activation functions, such as ReLu [43], introduce nonlinearity, allowing the network to capture complex patterns. At the final stage, FC layers aggregate the extracted features into predictions, enabling tasks such as classification or segmentation. CNNs use three-dimensional data for image classification: *width*, *height*, *depth*. As an example, an image from the CIFAR-100 dataset has dimensions $32 \times 32 \times 3$, corresponding to width, height, and 3 color channels (RGB). The following provides an overview of function of the layers.

Convolutional Layer The convolutional layer is the foundation of CNNs. It applies a set of filters (called kernels) to the input image, performing convolutions to produce feature maps [14]. Each filter is designed to detect specific features, such as edges or textures. The filter, which has smaller dimensions than the input, is applied to an area of the image, computing dot products between the weights of the filter and the input pixels, as illustrated in figure 2.4. Afterwards, the filter shifts by a stride, repeating the process until the kernel has computed the dot products for entire image. This process enables the network to learn spatial hierarchies of features, with the early layers capturing patterns like edges, and later layers capturing detailed structures [44]. After each convolution, the ReLu transformation is applied to introduce non-linearity [14].

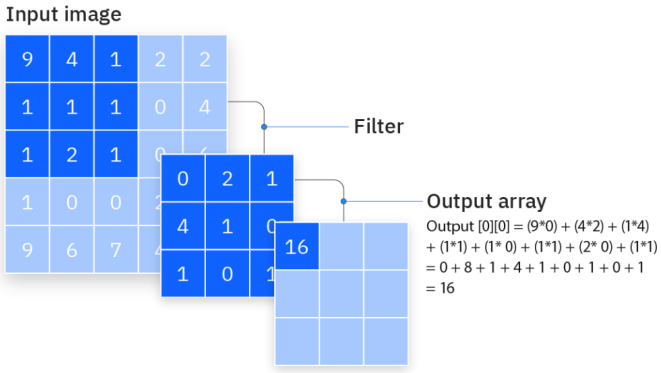


Figure 2.4: Illustration of a convolution with a 3×3 kernel [45].

Pooling Layer The function of the pooling layer (also known as downsampling) is to reduce the spatial dimensions of the feature maps, decreasing the number of parameters and computational load of the netwrok [14]. Moreover, this contributes to making the detection of features invariant to scale and orientation.

Fully Connected Layer The FC layer, as the name implies, connects every neuron in one layer to every neuron in the next. Positioned at the end of the network, this performs the task of classification based on the features extracted from the previous layers [14]. The raw output of the FC layer are called logits [46], and refers to the K-dimensional output vector for K classes. These have raw outputs can range from $[-R, R]$, where R are real numbers. However, an activation function must be applied to the logits to get the final probabilities of the model prediction. For multi-class classification, this activation function is the Softmax function introduced later in section 2.4.

These layers, combined with the architectural design of CNNs, introduce inductive biases that make them particularly well-suited for image-related tasks [47].

CNNs gained popularity after the introduction of LeNet-5 by LeCun et al. in 1998 [40], which showcased their capability to recognize handwritten digits. The field advanced significantly in 2012 when AlexNet [41] won the ImageNet

Challenge [48], demonstrating the effectiveness of deeper architectures and multi-GPU training for large-scale image recognition tasks. Subsequent developments led to influential architectures such as VGGNet [49], GoogLeNet [50], and the state-of-the-art ResNet [3]. The CNN architectures explored in this thesis are detailed in later in this section.

2.2.3 Vision Transformers

ViTs, introduced by Dosovitskiy et al. in 2021 [4], presented an alternative way of handling image classification by leveragering the transformer architecture. Transformers, introduced by Vaswani et al. in 2017 [51], was designed for Natural Language Processing (NLP), and revolutionized the deep learning field, surpassing Recurrent Neural Networks (RNNs) in NLP tasks [52, 51]. The transformer architecture is based on self-attention mechanism, allowing the model to selectively weigh the significance of each part of the input. As their design lack recurrence or convolutions, transformers use positional embeddings to represent the order of tokens in a sequence.

Likewise, in ViTs, images are represented as sequences including the class label as a learnable token for classification. To start, the input image is divided into a sequence of patches, which are then flattened and linearly embedded into a vector. The spacial information is preserved by adding positional encodings to the embeddings. Next, the sequence is fed into a transformer encoder identical to that introduced by Vaswani et al., consisting of alternating layers of Multi-head self-attention (MSP) and Multi-Layer Perceptron (MLP) blocks with Layer Norm (LN) applied before every block and residual connection after every block. The final MLP layer acts as the classification head. The illustration in figure 2.5 shows the architecture of Vision Transformers.

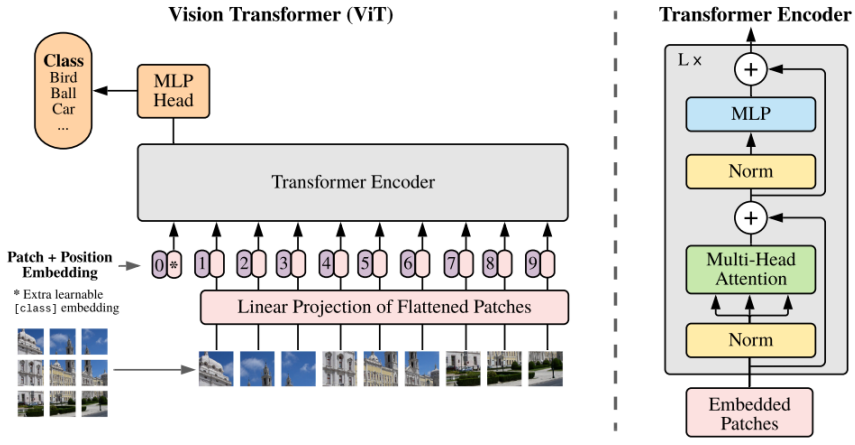


Figure 2.5: Vision Transformer architecture [4]. The input image is split into fixed-sized patches, each liniearly embedded with added postion embedding. These are then fed to a transformer encoder.

Unlike CNNs, ViTs have much less image-specific inductive bias [4]. While the inductive bias in CNNs help the model generalize to unseen data [47], the ViT

architecture does not include built-in assumptions about the structure of the data, and only the MLP layers are local an translationally equivalent [4]. As a result, ViTs require more training data to learn spacial relations compared with CNNs. However, ViTs differ from CNNs by applying self-attention mechanism across the entire image, enabling the model to capture global dependecies by attending to any patch of the image at any layer, whereas CNNs applies convolution operations over receptive fields, evolving through stacking layers [14].

2.2.4 ResNet-50 Architecture

ResNet-50 is a variant of the Residual Network (ResNet) architecture, developed by Microsoft Research (He et al.) in 2015 [3]. The ResNet architecture was designed to address the vanishing and exploding gradient problem in deep networks. When training a deep neural network, as the number of layers increases, the gradients of the loss function with respect to the weight can potentially vanish or explode during backpropagation [3]. This can lead to slow convergence or unstable updates.

In traditional CNNs, stacked layers learn a direct mapping $\mathcal{H}(x)$ of the input x to the output. The introduction of residual layers, however, allows for the layer to fit a residual mapping, as illustrated in figure 2.6, where the network learns the residual function $\mathcal{F}(x) = \mathcal{H}(x) - X$. Isolating $\mathcal{H}(x)$ yields $\mathcal{H}(x) = \mathcal{F}(x) + x$, meaning that the input x is passed through a skip connection. This architecture reduces the complexity of the optimization process, as the network only has to model the residual component $\mathcal{F}(x)$ rather than the full mapping $\mathcal{H}(x)$ [3].

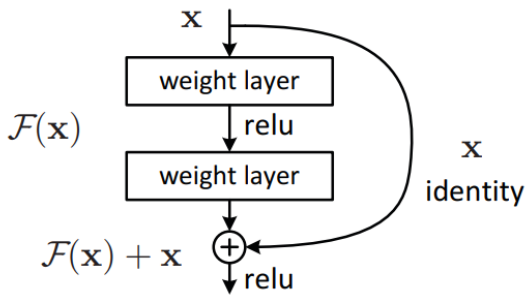


Figure 2.6: Residual learning block [3].

The residual network is formed by stacking multiple layers of residual blocks, as the one depicted in figure 2.6. The ResNet architecture can have a varying number of layers, and, as the name implies, the ResNet-50 variant has 50 layers. Other architectures include ResNet-34, ResNet-101, and ResNet-152 [53].

The ResNet-50 implementation consist of 1 convolutional layer, 4 stages of bottleneck residual blocks with layers [3, 4, 6, 3], global average pooling, and finally a fully connected layer for classification [54], as seen in figure 2.7.

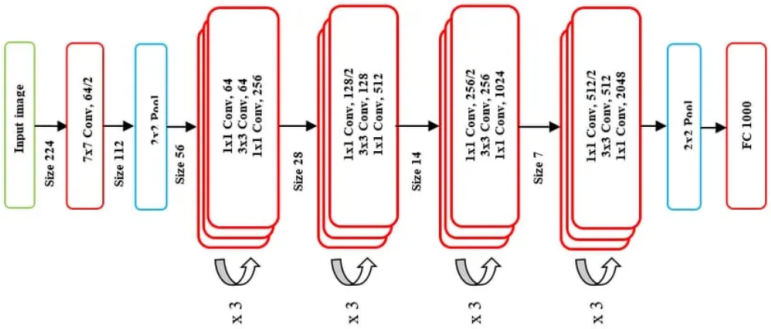


Figure 2.7: ResNet-50 architecture [55], consisting of 6 stages: input preprocessing, 4 blocks with residual connections, and a FC layer.

2.2.5 MobileNetV2 Architecture

MobileNetV2, introduced by Sandler et al. [10], 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 [56], 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, as described above, allow for identity mapping and improved gradient flow, MobileNetV2 employs an inverted residual structure [10]. 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 2.8, helps preserve crucial information and maintain a rich feature space without substantially increasing computational cost. The use of a linear bottleneck (i.e., no nonlinear activation in the low-dimensional projection) also helps prevent the destruction of useful information, further improving efficiency and accuracy.

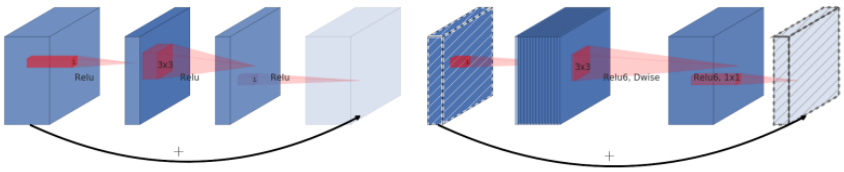


Figure 2.8: Residual (left) and inverted residual rlocks (right) [10]. The thickness of the blocks indicate their relative number of channels.

Depthwise Separable Convolution Following MobileNetV1, MobileNetV2 relies on depthwise separable convolutions, depicted in Figure 2.9, to factorize the convolution operation into two simpler operations [56]: depthwise convolution and pointwise convolution. 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 filters 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 [56, 10].

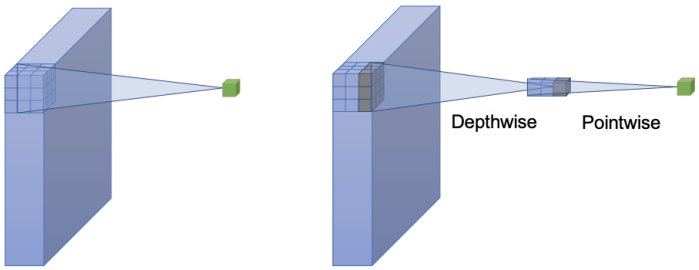
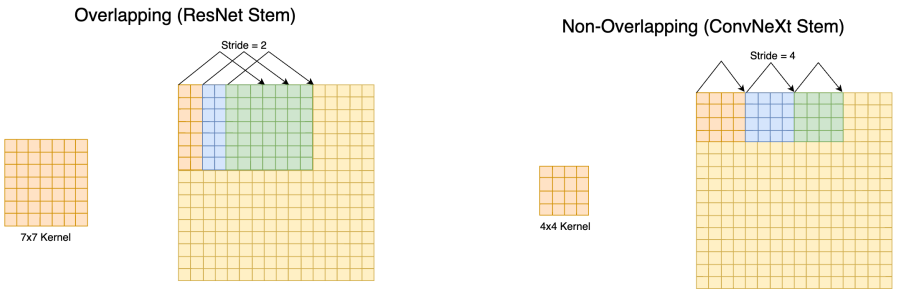


Figure 2.9: Standard convolution (right) and depthwise separable convolution (left) [57].

2.2.6 ConvNeXt Base Architecture

ConvNext, introduced by Liu et al. in 2022 [11], evolutionized the traditional CNN architecture by incorporating elements from ViTs. Starting with the stem cell, the ConvNeXts employ a patchify stem, as seen in figure 2.10. By setting the kernel to 4×4 and the stride to 4, the result is a non-overlapping convolution where no information is shared between them, yet these are later combined in the final layers of the network.



(a) (b)

Figure 2.10: ResNet stem (a) vs ConvNeXt stem (b). Figure modified from [58].

Next, inspired by ResNeXt [59], the ConvNeXt use depthwise convolution, already described in Section 2.2.5, to increase the computation vs. accuracy trade-off. Likewise, the ConvNext incorporates inverted bottleneck blocks, which are an

important design in transformers as well as MobileNetV2 [11]. Additionally, the ConvNext architecture uses a 7×7 depthwise convultion in each block on the condition of moving the depthwise convolution layer up. See figure 2.11.

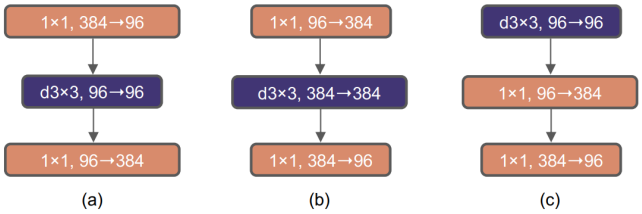


Figure 2.11: Block modifications [11]. Here, (a) is a ResNeXt block, (b) is an inverted bottleneck block, and (c) shows the position of the spatial depthwise convolutional layer moved up.

Instead of using the ReLu activation function, commonly used in ResNets, the ConvNeXt use Gaussian Error Linear Unit (GeLU) [60]. This activation function has shown to perform better than ReLU for transformers [11]. Additionally, ConvNext draws inspiration from transformers by removing most instances of normalization layers, only leaving them after depthwise convolutions and furhter replacing the Batch Normalization (BN) with LN. Finally, ConvNeXt introduces separate downsampling layers between blocks rather than implementing downsampling within the blocks themselves.

These implementations has shown to improve performance, competing with ViTs [11].

2.2.7 ViT-B/16 Architecture

The ViT-B/16 [4] is a ViT that leverages the transformer architecture for image classification, as described in Section 2.2.3. One of the main characteristics of the ViT-B/16 is that the input image consist of a fixed 16×16 patch size. It has 12 stacked encoder layers, each computing self-attention with 12 heads. Additionally, each layer includes an MLP. Skip-connections are applied around self-attention and MLP blocks for better gradient flow. LN is applied before the attention and MLP blocks. Afterwards, the appended class token is extracted, and a fully connected layer maps the token to the class prediction [61].

2.3 Class Re-balancing Methods for Long-Tailed Learning

Class re-balancing methods in deep learning seeks to mitigate the effects of imbalanced class distributions in training data. According to Zhang et al. [9] class re-balancing can be divided into three sub-categories: re-sampling, class-sensitive learning, and logit adjustment. In this thesis, the primary focus is on class-sensitive learning, exploring re-weighting, re-margining, and logit adjustment strategies. For completeness, re-sampling is briefly introduced as well.

2.3.1 Re-Sampling

Deep networks are commonly trained with mini-batch gradient descent using random sampling, i.e. each sample has an equal probability of being selected [9]. When classes are imblanced, samples from head classes naturally occur more often, thus having higher chances of being selected than samples from tail classes, resulting in a bias towards head classes. Re-sampling addresses this problem by adjusting the sample probabilites or the sample counts per class (e.g. oversampling minority classes or undersampling majority classes). While this approach seems simple and intuitive, re-sampling can lead to overfitting of tail classes or underperformance on head classes, hence the need for more sophistacated approaches [9].

2.3.2 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 [9], as described later in this section. 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 performance on tail classes.

The three sub-categories of class-sensitive learning explored in this thesis are described in the following.

Re-Weighting

Modifies the loss function, commonly the cross-entropy loss, by assigning different weights to each class [9]. Classes with fewer samples are assigned greater weights, thus re-balancing the training process.

Re-Margining

Re-margining aims to solve class imbalance by introducing margin between classes, encouraging higher margins between rare classes.

Figure 2.12 illustrates how the margin affects the decision boundary for binary classification. Here, the margin γ_i of the i-th class is the minimum distance of the data in the i-th class to the decision boundary. By increasing the margin, the decision boundary separates the regions of input space corresponding to different class predictions, thus the model becomes more confident in classification [13].

Cao et al. [13] has shown that the optimal margin for a class is inversely proportional to the fourth root of its frequency n_i, allowing the model to generalize better in imbalance settings:

$$\gamma_i \propto n_i^{-1/4} \tag{2.3}$$

with the standard solution being $\gamma_1 = \gamma_2$.

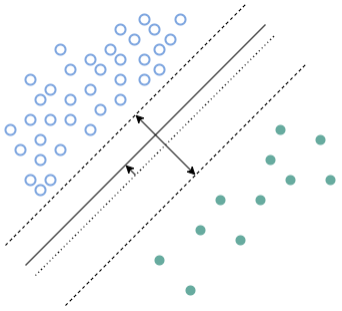


Figure 2.12: The margins and decision boundary of binary classification. The distance between the classes is fixed, but shifting the decision boundary can optimize the trade-off between them. [13].

Logit Adjustment

Logit adjustment is a class re-balancing technique that aims to optimize the class imbalance by adjusting the model outputs, i.e. logits, based on prior class probabilities [23, 17].

In classification tasks, model outputs η_j for class j are computed by:

$$\eta_j = \theta_j^T f(x) \tag{2.4}$$

where θ_j are the weights for the final layer for class j, and $f(x)$ is the feature extractor function [17]. Logit adjustment modifies these logits accounting for the prior class probabilites, thus shifting the model's prediction toward a more balanced output distribution [17].

In the following section, the theory behind representative loss functions commonly used in class-sensitive learning is presented, beginning with an introduction to loss functions followed by the baseline CE loss.

2.4 Loss Functions

Loss functions are a fundamental component in deep learning, as they serve as a measure of how far model predictions deviate from the actual values [2, 37].

During training, an optimization algorithm iteratively updates the model parameters (weights and biases) with the goal of minimizing the loss using the gradient of the loss function with respect to each parameter to determine how the parameter affects the loss [2]. Consequently, the loss function influences the model's interpretation of prediction errors and guides parameter updates. This process continues until the model converges or meets a stopping criterion. The experiments in this thesis are conducted using the Adam optimizer introduced by Kingma et al. [62].

For image classification tasks, the standard loss function is the cross-entropy loss combined with the softmax activation function [37] described in the following.

2.4.1 Softmax Activation Function

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 is defined as:

$$P(y = i \mid \mathbf{z}) = \frac{\exp(z_i)}{\sum_{j=1}^{K} \exp(z_j)} \tag{2.5}$$

These probabilities $P(y = i)$ sum to 1, and allows the network's outputs to represent the model's confidence in each class.

2.4.2 Cross-Entropy Loss

The CE loss is a fundamental building block in training deep classifiers and is widely regarded as the baseline in classification tasks [9, 14, 63]. The CE loss combines the cross-entropy loss and the softmax activation function, and thus is commonly referred to as the *Softmax Loss* [37, 2, 9].

The CE loss measures the difference between the predicted probability distribution P and the true class label y. For a single sample, the loss is defined as [37, 14]:

$$\mathcal{L}_{\text{CE}} = -\sum_{i=1}^{K} y_i \log(p_i) \tag{2.6}$$

where y_i is the ground truth label, represented as a one-hot encoded vector, and p_i is the predicted probability for class i. This simplifies for a single ground truth class y:

$$\mathcal{L}_{\text{CE}} = -\log(p_y) \tag{2.7}$$

Inserting the softmax probabilites from Eq. (2.5) yields:

$$\mathcal{L}_{\text{CE}} = -\log\left(\frac{\exp(z_y)}{\sum_{j=1}^{K} \exp(z_j)}\right) \tag{2.8}$$

where y is the true class. 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.

The standard CE loss (eq. (2.8)) estimates class probabilities based on the assumption that the training and validation labels follow the same distribution, $p(y = j)$, in other words how common is the label j in the data [17]. When training with an imbalanced dataset the estimated probabilities ϕ can become biased when applied to data with a different class distribution, e.g. a balanced test set, which can lead to errors in predictions.

2.4.3 Weighted Softmax Cross-Entropy Loss

The WCE loss modifies the standard CE loss (Eq. (2.8)) by introducing a weighting factor w. For multiclass classification, the CE loss it is multiplied by the inverse class frequencies $1/\pi_y$ [9, 15], given by:

$$\mathcal{L}_{\text{WCE}} = -w \log(p_y) = -\frac{1}{\pi_y} \log(p_y) \tag{2.9}$$

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

2.4.4 Focal Loss

FL [15] 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. (2.8)). Here, $\gamma \geq 0$ is a tunable focusing parameter. The FL modifies the CE loss by applying the inverse prediction probability as follows:

$$L_{fl} = -(1 - p_y)^\gamma \log(p_y) \tag{2.10}$$

This factor increases the weight of misclassified examples, ensuring the model prioritizes learning from challenging samples. Figure 2.13 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 2.13 represents the standard cross-entropy loss for $\gamma = 0$.

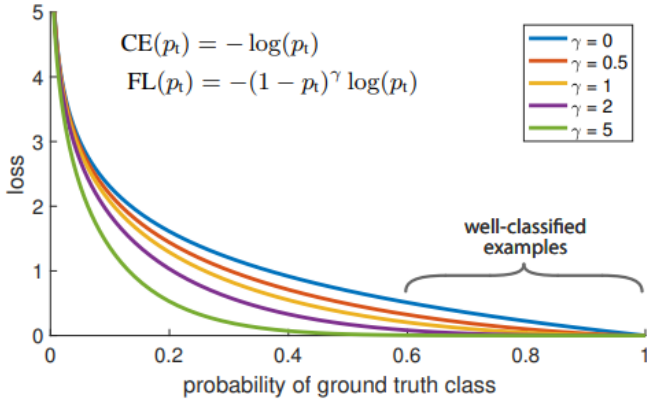


Figure 2.13: Focal loss for various values of γ. [15].

2.4.5 Class-Balanced Loss

CB loss [16] 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 2.14, 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 $(1 - p)$.

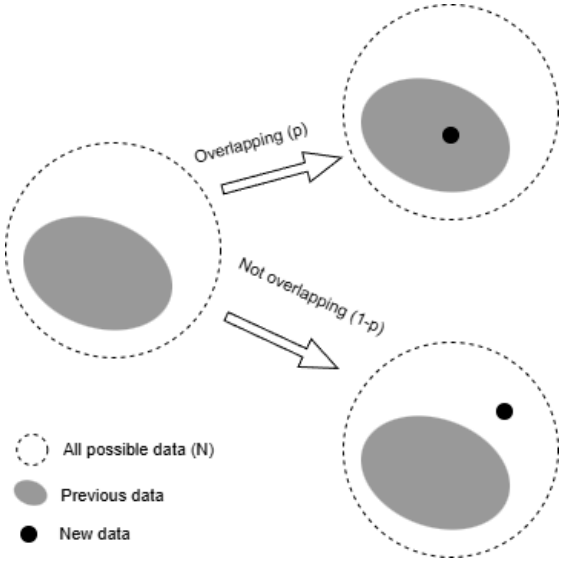


Figure 2.14: The probability of newly sampled data (black dot) to overlap with previously samples data (grey area) given the set of all possible data (dashed circle). [16].

Cui et al. [16] 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:

$$E_n = \frac{1 - \beta^n}{1 - \beta} \tag{2.11}$$

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 Eq. (2.8) yield the following:

$$L_{cb} = -\frac{1 - \beta}{1 - \beta^{n_y}} \log(p_y) \tag{2.12}$$

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 the total loss, as the effective number in Eq. (2.11) 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.

2.4.6 Softmax Equalization Loss

Equalization Loss (EQL) [18] aims to mitigate the over-suppression of tail classes, which occurs when 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. An analysis of gradient impact on classes is seen in figure 2.15.

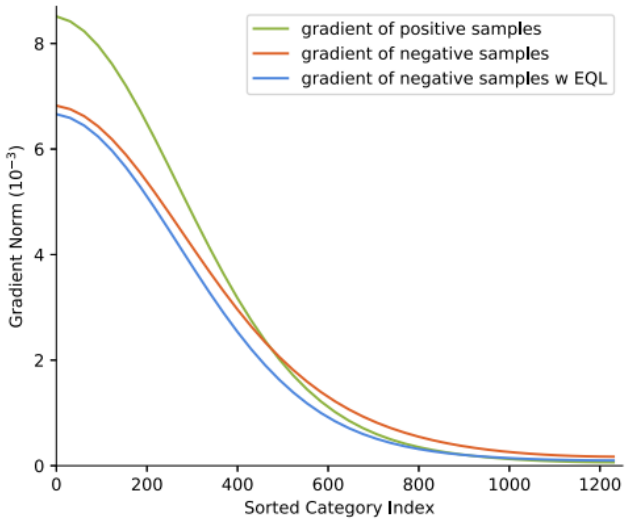


Figure 2.15: Gradient analysis on positive and negative examples along with negative samples with Equalization Loss [18].

The EQL 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:

$$\mathcal{L}_{SEQL} = -\sum_{j=1}^{C} y_j \log(\tilde{p}_j) \tag{2.13}$$

where

$$\tilde{p}_j = \frac{e^{z_j}}{\sum_{k=1}^{C} \tilde{w}_k e^{z_k}} \tag{2.14}$$

and the weight term is given by:

$$\tilde{w}_k = (1 - \beta T_\lambda(f_k))(1 - y_k) \tag{2.15}$$

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. As there is no background category for image classification tasks, β is introduced to randomly maintain the gradient of negative samplers. β is a random variable with probability γ of taking value 1 and probability $1 - \gamma$ of taking value 0.

2.4.7 Balanced Softmax Loss

The *Balanced Softmax (BS)* Loss modifies the standard softmax loss (eq. (2.8)) by directly incorporating class priors π_y into the logits before computing probabilities [17]. 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.

Ren et al. [17] introduces Softmax regression as a multinomial distribution, ϕ, dependent on η, defined as:

$$\phi_j = \frac{e^{\eta_j}}{\sum_{i=1}^{k} e^{\eta_i}} \tag{2.16}$$

where k are the number of classes.

The Balanced Softmax uses the output logits η (Eq. (2.4)) to parametize two separate probabilities: ϕ for testing, and $\hat{\phi}$ for training. This separation is used when training with an imbalanced dataset to eliminate the discrepancy between training and testing distributions [17].

Assume ϕ to be the desired conditional probability of the balanced dataset, with the form $\phi_j = p(y = j \mid x) = \frac{p(x|y=j)}{p(x)} \cdot \frac{1}{k}$, and $\hat{\phi}$ to be the desired conditional probability of the imbalanced training set, with the form $\hat{\phi}_j = \hat{p}(y = j \mid x) = \frac{p(x|y=j)}{\hat{p}(x)} \cdot \frac{n_j}{\sum_{i=1}^{k} n_i}$. If ϕ is expressed by the standard Softmax function of the model output η, then $\hat{\phi}$ can be expressed as:

$$\hat{\phi}_j = \frac{n_j e^{\eta_j}}{\sum_{i=1}^{k} n_i e^{\eta_i}}. \tag{2.17}$$

where n_j is the number of samples in class j.

Ren et al. [17] proved that this method can accomodate the label distribution shifts between training and test sets [17]. Ultimately, the Balanced Softmax loss function is becomes:

$$\mathcal{L}_{bs} = -\log\left(\frac{\pi_y \exp(z_y)}{\sum_j \pi_j \exp(z_j)}\right) \tag{2.18}$$

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.

2.4.8 LDAM Loss

The *Label-Distribution-Aware Margin (LDAM)* Loss [13] represents a class-sensitive approach based on re-margining. Instead of altering the loss directly, LDAM modifies the margin applied to each class' decision boundary, ensuring that the model is more confident in classification for minority classes. The goal is to apply the optimal boundary in Eq. (2.3) by increasing the distance between the largest and second largest logit for each class and introducing that to the cross-entropy loss (eq. (2.8)).

The class-dependent margin for multiple classes is defined as:

$$\gamma_j = \frac{C}{n_j^{1/4}} \tag{2.19}$$

where C is a tunable hyper-parameter, and n_j is the number of samples in class j. Applying the margin through the loss function, inspired by the Hinge Loss [13], will encourage the model to respect the margins in Eq. (2.19):

$$\mathcal{L}_{ldam-hg} = \max\left(\max_{j \neq y}\{z_j\} - z_y + \Delta_y, 0\right) \tag{2.20}$$

where

$$\Delta_j = \frac{C}{n_j^{1/4}} \quad \text{for } j \in \{1, \ldots, k\}. \tag{2.21}$$

and z_y is the model output (logit) for the true class y, and $\max_{j \neq y}\{z_j\}$ is the highest score among incorrect classes. Adding the margin Δ_j ensures that the true class logit is separated from the highest incorrect score by the at least Δ_j. However, since the function in Eq. (2.20) is non-differentiable at certain points, a modifications to the cross-entropy loss is used to address this. Here, the class-dependent margins are introduced directly into the logits as follows:

$$L_{ldam} = -\log\left(\frac{\exp(z_y - \Delta_y)}{\sum_j \exp(z_j - \Delta_j)}\right) \tag{2.22}$$

Chapter 3

Methodology

This chapter describes the methods and approaches used in the experiments. This includes the selection of dataset, models, loss functions, as well as training 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 [12] 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 [9]. 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

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 [7], Places365-LT [7], CIFAR-100-LT [13], and iNaturalist 2018 [8]. 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 [7]	1,000	115,846	50,000
CIFAR-100-LT [13]	100	50,000	10,000
Places-LT [7]	365	62,500	36,500
iNaturalist 2018 [8]	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. Its 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 compared to larger dataset, like ImageNet-LT or iNaturalist. For example, CIFAR-100 images are of relatively low resolution (32×32 pixels), as depicted in Figure 3.1, 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.

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 [9], which serves as a reference for creating a similar distribution across the CIFAR-100-LT training, evaluation, and test datasets.

Figure 3.1: Examples from the CIFAR-100 dataset showing the low resolution (32×32 pixels). Each image is labeled with its corresponding class name.

The ImageNet-LT dataset, introduced by Liu et al. [7], has been widely used in empirical studies, including *Deep Long-Tailed Learning: A Survey* [9]. The class distributions of its training set is plotted in figure 3.2. The ImageNet-LT is constructed from the original ImageNet-2012 following the Pareto-distribution (Eq. (2.2)) with power value $\alpha = 6$ [7], 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 3.2, the most frequent class has 1280 images, and the least frequent class 5 images [7]. The power value of 6 is relatively large, resulting in the steep imbalance and very few samples in the tail, as seen in figure 3.2. 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 with 20 samples per class in the validation set, and 50 samples per class in the test set.

However, investigating the CIFAR-100-LT dataset used by Cao et al. [13] 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 Eq. (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 3.3. The following section will describe the preparation of CIFAR-100 for the experiments in this thesis.

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 with PyTorch [64]. The training and test datasets were preprocessed by converting the images to tensors using the ToTensor

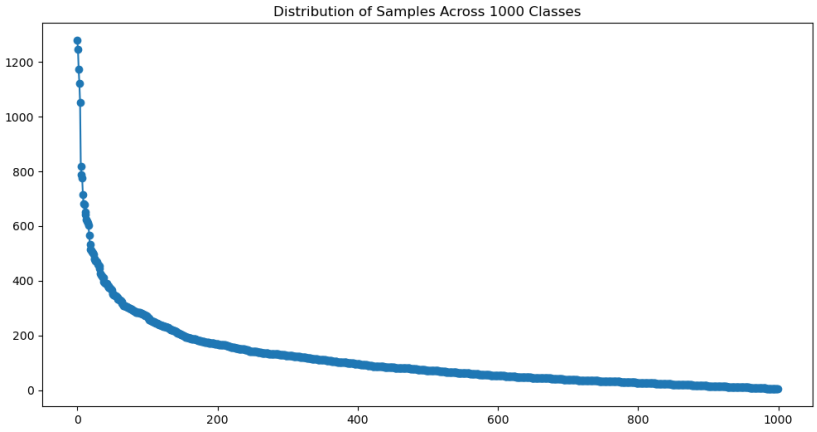


Figure 3.2: The class distribution of the training images for the ImageNet-LT dataset shows a long-tailed distribution.

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.

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	9,754
Long-Tailed Test Set	1,051
Head	844
Middle	169
Tail	38

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 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 [13], the imbalance was created using exponential decay. Here, the number of samples per class decreases exponentially, controlled by the imbalance factor, following Eq. (2.1). For this thesis,

an imbalance factor of 100 was applied. This means that the most frequent class contains a 100 times more samples than the least frequent class.

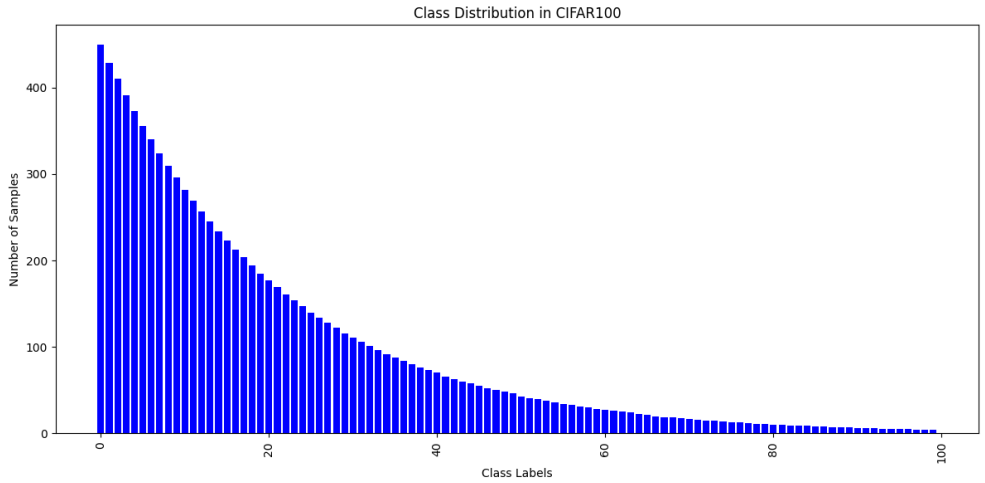


Figure 3.3: The class distribution of the CIFAR-100 long-tailed training set with 450 samples in the most frequent class.

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 3.3. 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 3.4. The dataset was saved locally to maintain reproducibility.

To further analyze performance, the test set was divided into three subsets: head, middle, and tail classes, comprising of 1/3 each (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] [13]. 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

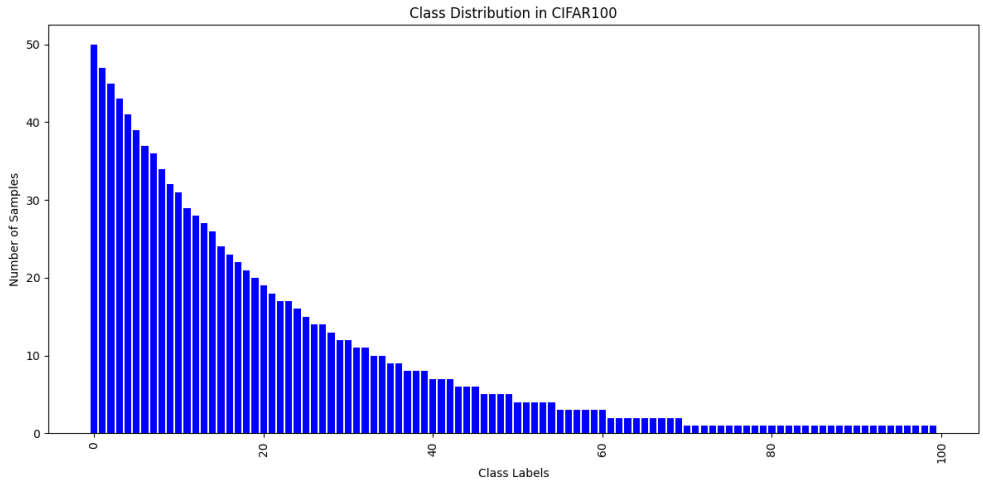


Figure 3.4: The class distribution of the CIFAR-100 long-tailed test set with 50 samples in the most frequent class.

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

This section outlines the methods employed to address the challenges posed by a long-tailed data distribution. Specifically, it discusses the choice of model architecture and class re-balancing methods, including their strengths, limitations, and rationale based on prior research and feasible implementations.

3.3.1 Model Selection

This section will discuss the choice of model architectures addressing the challenges of long-tailed datasets. The models evaluated in this thesis are four common state-of-the-art (SOTA) models: ResNet-50, MobileNetV2, ViT-B/16, and ConvNeXt-Base. These SOTA models were selected to represent a diverse range of architectures for solving image classification tasks and to pose a balance between computational efficiency and state-of-the-art performance. The models were modified and trained using transfer learning with pre-trained weights from ImageNet

[48]. Model complexity can be viewed in table 3.3.

For all models, the fully connected layer is modified to accomodate for the CIFAR-100 dataset's 100 classes.

Transfer learning leverages pre-trained neural networks to adapt to new tasks on smaller datasets, a method successfully applied in domains such as medical imaging and environmental monitoring [65, 66, 67]. By utilizing models pre-trained on large-scale datasets like ImageNet, transfer learning allows for the retention of generalized features, enabling high performance even with limited data. In the context of this project, transfer learning is employed to evaluate its effectiveness in addressing the challenges of long-tailed datasets, specifically by enabling the model to generalize well across classes with varying sample sizes.

Table 3.3: Overview of the models used in experiments, including architecture type, pretraining dataset, parameter count, and top-1 accuracy on pretraining data.

Model	**Type**	**Pre-trained**	**Parameters (millions)**	**top-1 accuracy**
MobileNetV2 [10]	CNN	ImageNet-1K	2.35	72.154 [68]
ResNet50 [3]	CNN	ImageNet-1K	23.71	80.858 [69]
ConvNeXt-Base [70]	CNN	ImageNet-1K	87.67	84.062 [54]
ViT-B/16 [4]	ViT	ImageNet-21K (ft on ImageNet-1K)	85.88	-

ResNet-50 Resnet-50 [3] is a convolutional neural network designed with residual connections to mitigate the vanishing gradient problem, as detailed in section 2.2.4. Introduced by He et al. in 2015, the ResNet architecure achieved unprecedented results and won ILSVRC 2015 for image classification tasks [48].

With 50 layers, ResNet-50 strikes a balance between model complexity and computational efficiency. It is highly effective for extracting robust features, making it well-suited for image classification on long-tailed datasets [3]. ResNet-50 has been utilized in various domains of image classification, including tasks such as medical imaging [71, 72] and face detection [73], and commonly serves as a baseline in research [28, 74, 27, 23]. Leveraging pretrained weights on ImageNet-1K, it achieves better generalization on unseen datasets [75, 76, 77, 78], with its top-1 accuracy shown in Table 3.3.

Compared to smaller architectures like ResNet-16 and ResNet-34, ResNet-50 offers improved feature extraction while being more memory-efficient and faster to train than larger versions, such as ResNet-101 and ResNet-152 [3]. This balance of performance and efficiency makes it suitable for smaller datasets like CIFAR-100 [79].

Chosen as a reference point for CNN designs in this study, ResNet-50 demoonstrate the advantages of transfer learning and remains a reliable choice for addressing class imbalance effectively. Its demonstrated versatility across various datasets underscores its value for robust image classification tasks [3, 28].

MobileNetV2 The MobileNetV2 model is a successor to the ResNet architecture, adjusted to run on mobile devices. It requires fewer computational resources due to the introduction of inverted residual blocks and depthwise separable convolutions, as described in section 2.2.5. This architecture was chosen to represent an example of a modern and lightweight CNN architecture that has achieved state-of-the-art performance [10], and has shown great performance in image classification tasks, including medical imaging [80] and fruit classification [81, 82]. Its lightweight design makes it particularly attractive for applications where computational resources are limited. By including MobileNetV2 among the evaluated architectures, the performance is compared across varying complexity levels, providing insights into how efficiency-oriented designs fare against more complex models. This comparison is especially relevant if the application domain involves real-time processing or deployment on mobile or embedded devices [10].

ViT-B/16 The Vision Transformer (ViT-B/16) [4] represents a paradigm shift in image classification by replacing convolutional operations with a transformer-based architecture. Unlike convolutional neural networks, ViTs treat images as sequences of patches, enabling the model to capture global relationships across the entire image, as described in Section 2.2.3. ViT-B/16 divides images into 16×16 patches, which are then flattened and embedded before being processed by the transformer encoder.

Its reliance on self-attention mechanisms allows it to effectively model long-range dependencies, making it particularly suitable for complex image classification tasks. However, its data-hungry nature means that it benefits significantly from pretraining on massive datasets [4].

The ViT-B/16 model was implemented using the timm library [83], pretrained on ImageNet-21K and fine-tuned on ImageNet-1K. For consistency, and a more direct comparison with the other models, it would have been preferable to use the PyTorch library implementation [61], which is pretrained solely on ImageNet-1K. Considering none of the layers are frozen during training, allowing for fine-tuning on the CIFAR-100 dataset, this choice has potentially influenced the model performance, despite the fact that ViTs benefit from pretraining on large-scale datasets due to the lack of inductive bias (see Section 2.2.3) [4, 84] .

In the context of this project, ViT-B/16 was selected to evaluate the performance of transformer-based architectures on long-tailed datasets. Its ability to generalize across diverse visual features provides a valuable comparison to CNN designs, as the ViT outperformed state-of-the-art CNNs when pre-trained on large datasets [4], and has been used in image classificataion tasks, such as brain tumor classification [85]. Additionally, ViT-B/16's flexibility in capturing non-local dependencies offers potential advantages in addressing class imbalance by leveraging the full spatial context of underrepresented classes.

By including ViT-B/16 in the model evaluation, this study explores the applicability of transformer-based architectures in scenarios involving limited data or significant imbalance, providing insights into their effectiveness relative to convolutional counterparts.

ConvNeXt Base ConvNeXt Base [11] represents a modern CNN architecture inspired by ViTs. It incorporates architectural enhancements such as depthwise convolutions, inverted bottlenecks, and improved normalization techniques, while reatining the simplicity of convolutional operations, as discussed in section 2.2.6. These innovations allow ConvNeXt to achieve state-of-the-art performance in image classification tasks while retaining the core efficiency of convolutional operations [11]. In its original paper [11], ConvNeXt Base outperformed many traditional CNN architectures on ImageNet-1K, achieving competitive accuracy while maintaining computational efficiency.

Compared to smaller variants of ConvNeXt (Tiny, Small), ConvNeXt Base has a higher capacity for feature representation due to its larger parameter count and deeper architecture (see table 3.3). This is particularly advantageous for capturing the complexities of long-tailed datasets where subtle differences in tail classes require richer feature extraction [11]. ConvNeXt Base offers competitive performance while being significantly more efficient in terms of computational resources and training time, whereas larger variants (Large, XL) require considerably more memory and computational power, which might not be justified given the scale of the CIFAR-100 dataset. Additionally, for datasets like CIFAR-100, which have relatively low-resolution images, using very large models may lead to diminishing returns, as the dataset's complexity might not fully leverage the capacity of larger architectures.

3.3.2 Class-Sensitive Methods

This thesis investigates the interplay between model architectures and class-sensitive loss functions in addressing long-tailed image classification tasks. The selected architectures encompass diverse designs tailored for image classification, while the chosen loss functions span a range of methods designed to mitigate class imbalance. Drawing from the cost-sensitive approaches discussed in Deep Long-Tailed Learning: A Survey [9], this section outlines the rationale behind selecting suitable loss functions for long-tailed distributions, beginning with the standard softmax loss as a foundational baseline.

Softmax Loss Conventional training of deep networks typically relies on the Softmax Cross-Entropy (CE) loss [9], which serves as a foundational baseline in this study. However, this loss inherently favors head classes as these contribute not only more positive examples but also a disproportionately large number of negative examples for other classes, amplifying their influence on the gradient updates. In contrast, tail classes, with fewer positive samples, are underrepresented in the training process and more frequently suppressed [9, 15], as discussed in section 2.4. This imbalance skews the model's decision boundaries, resulting in strong performance on head classes but poor perfomance on tail classes.

To mitigate these limitations, researchers have proposed class-sensitive loss functions designed to address various aspects of class imbalance. These methods can be broadly categorized into re-weighting, logit adjustment, and re-margining

approaches, each offering unique strategies to balance the training process. The following details these methods, briefly explaining their mechanisms and highlighting their benefits in the context of long-tailed image classification tasks.

Weighted Softmax Loss In scenarios of severe imbalance, minority classes are overshadowed by the abundant head classes under conventional training. Although simple, Weighted Softmax Loss (WCE) [9] provides a direct and intuitive method for re-weighting: each class's contribution to the parameter updates is modulated to reflect its scarcity in the dataset distribution, as discussed in section 2.3.2. This helps maintain a more balanced gradient flow during training, preventing head classes from overtaking the optimization. However, using the inverse class frequency may not always yield optimal results, and more sophisticated weighting methods can be employed [9]. 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. As it is easy to implement and understand, WCE is considered as an initial step towards handling imbalance before exploring more complex techniques. Overall, WCE exemplifies how re-weighting can effectively guide the model to learn less-represented categories more robustly, thereby improving long-tailed recognition performance.

Class-Balanced Loss The Class-Balanced Loss [16], like the WCE loss, applies a weighted term to the standard CE loss; however, it introduces a novel concept of effective number to approximate the expected number of samples per class, and thus avoiding each class to be scaled solely based on its frequency [9, 16]. 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, as discussed in section 2.3.2. This approach balances the training 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 [9]. As a result, CB Loss ensures that each class, regardless of its 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 while retaining head class accuracy [16].

Balanced Softmax Loss The Balanced Softmax Loss [17] addresses class imbalance differently from traditional re-weighting methods like Weighted Cross-Entropy (WCE) and Class-Balanced (CB) losses. Instead of adjusting the loss function after probability computation, it modifies the logits directly by scaling them with class frequencies. This adjustment shifts decision boundaries, preventing classes with fewer samples from being overshadowed by majority classes, as detailed in section 2.4.7.

By aligning the model's posterior distribution with true class priors during both training and inference, the Balanced Softmax Loss fosters a more uniform and accurate representation of all classes. This approach promotes balanced internal representations, improving the recognition of tail classes without disproportionately penalizing head classes. Empirical evidence suggests that this method can outperform other re-weighting strategies by addressing class imbalance at the logit level, effectively correcting the bias at its source [17].

Focal Loss Instead of using training labels, Focal Loss [15] modifies the CE loss by including a focusing parameter that down-weights well-classified examples, thus reducing the dominance of easily recognized samples. Consequently, tail classes, which inherently present more challenging classification problems, receive increased attention with the dynamic re-weighting term, as detailed in section 2.3.2. 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. Though originally introduced in the context of object detection, it has been widely adopted in image classification tasks with long-tailed distributions, and empirical results have shown that this approach can significantly boost performance on minority classes without severely degrading performance on majority classes [15, 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.

Equalization Loss In highly imbalanced scenarios, positive samples from head classes can lead to negative gradient over-suppression of tail classes [9]. Over time, this can discourage the network from correctly identifying the minority classes. Equalization Loss [18] counteracts this by selectively down-weighting the negative gradients that arise when tail classes appear as negative samples for majority classes, as detailed in section 2.3.2. This strategy highlights a subtlety in the re-weighting paradigm: it is not only about improving tail class importance, but also about preventing excessive suppression of these classes through negative gradients. However, excessively down-weighting negative gradients may impair the model's discriminative abilities [9]. EQ Loss exemplifies a more fine-grained approach to re-weighting, intervening at the gradient level to ensure fair treatment of minority categories.

LDAM Loss Re-margining attempts to modify the loss function to incorporate class-specific margins [9]. This adjustment aims to shift the decision boundaries farther from tail classes resulting in a different minimal margin between features and classifiers, as detailed in section 2.4.8. Label-Distribution-Aware Margin (LDAM) [13] incorporates class-dependent margins that are inversely related to class frequencies, providing larger margins to tail classes and smaller margins to head classes. 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. This

approach diverges from re-weighting methods, as re-margining directly influences how decision boundaries are formed in the embedding space, whereas re-weighting modifies the magnitude of the loss contributions. 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. 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 [13]. Combining LDAM with other techniques, such as re-weighting or re-sampling, can yield even better improvements, as each method addresses a complementary aspect of the imbalance issue [13].

Cao et al. [13] also introduced a complementary strategy known as Deferred Re-Weighting (DRW). In this approach, the training process begins without re-weighting to allow the model to learn robust feature representations, and re-weighting is applied only in later epochs. DRW ensures that the model first captures the overall data structure before tackling class imbalance, preventing the instability that early re-weighting can introduce.

While combining LDAM with DRW has been shown to enhance performance on imbalanced datasets, this thesis focuses solely on LDAM without integrating DRW [13]. The decision to omit DRW was made to isolate and evaluate the impact of re-margining alone, providing a clearer understanding of its effectiveness in addressing class imbalance.

The loss functions discussed in this section represent a broad spectrum of techniques addressing class imbalance, ranging from simple re-weighting to more sophisticated approaches like logit adjustment and re-margining. By incorporating these methods, this study aims to comprehensively evaluate their effectiveness in conjunction with diverse model architectures for long-tailed image classification.

Table 3.4 summarizes the loss functions employed for class-sensitive learning in this thesis, outlining their formulations and categorizations into re-weighting, re-margining, and logit adjustement strategies.

Table 3.4: Summary of losses. In this table z and p are the predicted logits. n are the number of samles, and n_y are the number of samples for class y. π denoted the vector of sample frequencies, where $\pi_y = n_y/n$ represents the label frequencies for class y. The class-wise weights are denoted w, and the class-wise margin is denoted Δ. γ denoted loss related tunable parameters.

Loss Name	**Formulation**	**Type**
Softmax loss [63]	$\mathcal{L}_{ce} = -\log(p_y)$	-
Focal loss [15]	$\mathcal{L}_{fl} = -(1-p_y)^{\gamma}\log(p_y)$	re-weighting
Weighted Softmax loss [9]	$\mathcal{L}_{wce} = -\frac{1}{\pi_y}\log(p_y)$	re-weighting
Class-balanced loss [16]	$\mathcal{L}_{cb} = -\frac{1-\gamma}{1-\gamma^{n_y}}\log(p_y)$	re-weighting
Balanced Softmax loss [17]	$\mathcal{L}_{bs} = -\log\left(\frac{\pi_y \exp(z_y)}{\sum_j \pi_j \exp(z_j)}\right)$	logit adjustment
Equalization loss [18]	$\mathcal{L}_{eq} = -\log\left(\frac{\exp(z_y)}{\sum_j \omega_j \exp(z_j)}\right)$	re-weighting
LDAM loss [13]	$\mathcal{L}_{ldam} = -\log\left(\frac{\exp(z_y-\Delta_y)}{\sum_j \exp(z_j-\Delta_j)}\right)$	re-margining

3.4 Training and Evaluation

All model architectures were trained with each loss configuration, and all combinations were trained on the CIFAR-100 dataset with 450 samples per class and evaluated on the balanced test set, the long-tailed test set, and the long-tailed subcategories; head, middle, and tail classes. The experimental results are presented and discussed in chapter 4.

3.4.1 Implementation Details

All models are trained using the ImageNet-1K pretrained weights from the torchvision library [54, 68, 86] except for ViT-B/16 that used the ImageNet-21K pretrained weights and ImageNet-1K fince-tuning from the timm library [83]. The fully connected layer was replaced with 100 classes to match the CIFAR-100 dataset. The specifications can be seen in table A.2. Each model is trained with a batch size of 128 and the Adam optimizer [62] with default settings for 90 epochs. The inizial learning rate is set to 0.001 which is reduced by a factor of 0.1 every 30 epochs using a StepLR scheduler [87]. The model checkpoint for the best validation accuracy is saved and loaded for testing. Experiments are conducted on four NVIDIA TITAN X (Pascal, 12 GB). See Appendix A for detailed experimental specifications.

Optimizer

The Adam optimization algorithm [62] is a popular optimizer in deep learning tasks, including image classification [88, 89]. Despite the Stochastic Gradient Desect (SGD) [90] optimizer being commonly used for training deep learning models, Adam was chosen for experiments in this study, as it requires less hyperparameter tuning compared to SGD, which often necessitates careful tuning of the learning rate and momentum to achieve optimal performance [62]. Adam combines the benefits of momentum and adaptive learning rates, enabling faster convergence and more stable training, which makes Adam suitable for experiments involving multiple loss functions and model architectures, as it provides a consistent baseline for comparison without introducing additional variables. However, studies have shown that training with SGD with momentum yield better performance on small datasets like CIFAR-100 [23]. Future work could include training with the SGD with fine-tuning of hyperparameters.

This study implements the PyTorch Adam optimizer [91] with default settings: $lr = 0.001$, $betas = (0.9, 0.999)$, $eps = 1e-8$, $weight\_decay = 0$, $amsgrad = False$.

Fine-Tuning

Transfer learning of weights from the larger ImageNet dataset can help overcome the problem of data scarcity in the long-tailed CIFAR-100 datasets [14, 92, 88]. Fine-tuning is an aspect of transfer learning, where parameters of a pre-trained

model are trained on the new target data. In this study, none of the layers in the pretrained models were frozen during training, resulting in fine-tuning of all parameters, allowing the model to fully adapt its learned feature representations to the CIFAR-100 dataset [92].

However, this approach comes with trade-offs. Fine-tuning all layers increases the risk of overfitting [14], especially when training on a smaller dataset. Additionally, the pretrained feature representations from ImageNet are optimized for general object recognition, and fully fine-tuning the network may lead to the loss of some of these general features learned from ImageNet [14]. This is due to the lower layers learning more generic features, like edges and shapes, and the top layers learn more specific features common for the dataset [44]. Future work could include freezing layers during training.

3.4.2 Performance Measures

The top-1 accuracy is a widely used benchmark evaluation metric in image classification tasks [9]. It measures the proportion of samples where the model's top prediction matches the ground truth label, and is given by [93]:

$$\text{Accuracy} = \frac{\text{correct classification}}{\text{all classifications}} \tag{3.1}$$

Due to the metric being well-suited for tasks with a single label corresponding to each input, the top-1 accuracy is used as the evaluation metric to measure the model performance on both balanced and long-tailed datasets in this thesis. Its role as a benchmark ensures compatibility with prior research, allowing direct comparisons of the results.

Performance under class imbalance is analyzed by calculating top-1 accuracy for subsets of the long-tailed test set, divided into head, middle, and tail classes. This separation of classes provides an insight into the model's ability to handle the challenges of long-tailed data, such as preserving high performance on tail classes while maintaining accuracy on head classes.

3.4.3 Baselines

Baselines are crucial when evaluating the performance of deep learning methods, especially in tasks involving class imbalance. In this thesis, the Softmax Cross-Entropy Loss, which is widely recognized for its simplicity and effectiveness in balanced image classificaion (see Section 2.4), serves as the primary baseline for both balanced and long-tailed training. This ensures a consistent benchmark for understanding the performance of the proposed long-tailed methods while also facilitating comparability with prior research.

Additional to the softmax baseline, all methods across all models are trained on the balanced CIFAR-100 dataset to establish performance baselines. This allows for direct comparisons of the results when trained on the CIFAR-100-LT data with the same models and loss functions. This setup represents the optimal training

conditions where all classes are equally represented, serving as a benchmark for the best achievable performance across all test sets.

Chapter 4

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.

4.1 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 ResNet-50 architecture (Acc1: 60.53%), closely followed by the ConvNeXt-Base architecture (Acc1: 57.89%). However, this improved tail-class performance comes at the cost of head-class accuracy, where ConvNeXt-Base outperforms ResNet-50 with a top-1 accuracy of 86.85% compared to 82.70%. Overall, ConvNeXt-Base demonstrates better performance across all classes (Acc1: 82.30%) on BS loss compared to ResNet-50 (Acc1: 79.16%). See Tables 4.2 and 4.8 for reference.

Remarkably, SEQ Loss consistently underperformed when trained with ResNet-50, MobileNetV2, and ViT-B/16, but achieved comparably good results with ConvNeXt-Base on tail classes (Acc1: 47.37%). 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 4.5), despite having the highest reported benchmark performance trained with balanced CIFAR-100 (Acc1: 93.95 %) among all model architectures investigated in this thesis. 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.

4.2 Overall Results

This section presents the overall results of all experiments conducted in this thesis, highlighting the best and worst performance of each loss design on specific models without directly comparing losses or models. All models are first trained on a balanced CIFAR-100 dataset, to provide a reference point for evaluating the effectiveness of class re-balancing strategies. It provides an overview of all findings to understand how each combination of architecture and loss function behaves, starting with the ResNet-50 architecture.

4.2.1 ResNet-50

From Table 4.1, Softmax and Weighted Softmax yield identical performance when trained on balanced data due to their cross-entropy term in equations (2.12) and (2.9). Balanced Softmax Loss demonstrates strong performance on the long-tailed test set (Acc1: 84.30%) and head classes (Acc1: 84.60%), while still performing comparably on tail classes. Notably, Sofmax Loss, Weighted Softmax Loss, and CB Loss all achieve top-1 accuracy of 94.74 % on tail classes as the greatest performance of all loss designs. Overall, the performance of the class re-balancing methods compare to that of the softmax baseline. The best performing loss design (Softmax) achieved and accuracy of 83.24%, not far behind the best published result for ResNet-50 with an accuracy of 86.90% as seen in table 4.9.

Table 4.1: 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	0.8144	0.8173	0.8199	0.7751	**0.9474**
Balanced Softmax loss	0.8310	**0.8430**	**0.8460**	0.8107	0.9211
Equalization loss	0.8316	0.8354	0.8365	**0.8166**	0.8947
LDAM loss	0.7990	0.7983	0.8069	0.7337	0.8947

From Table 4.2, Balanced Softmax Loss significantly improves tail-class accuracy (Acc1: 60.53%), outperforming other loss designs, 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. Comparing with the results form table 4.1, most losses exhibits strong performance on head classes, except for SEQL with an underwhelming performance across all categories. Likewise, LDAM struggles with most categories, and particularly with tail classes.

Table 4.2: 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	0.4970	0.7450	0.8069	0.5562	0.2105
Balanced Softmax loss	**0.5908**	0.7916	0.8270	**0.6568**	**0.6053**
Equalization loss	0.0886	0.1399	0.1505	0.1124	0.0263
LDAM loss	0.3742	0.5937	0.6469	0.4438	0.0789

4.2.2 MobileNetV2

Table 4.3: 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	0.8008	0.8049	0.8140	0.7456	0.8684
Balanced Softmax loss	0.8034	0.8030	**0.8069**	0.7574	**0.9211**
Equalization loss	**0.8078**	0.7916	0.7962	0.7396	**0.9211**
LDAM loss	0.7828	0.7821	0.7808	0.7574	**0.9211**

As shown in Table 4.3, SEQL achieves the highest accuracy on the balanced test dataset (Acc1: 80.78%) and, along with BS Loss and LDAM Loss, yields the best performance on tail classes (Acc1: 92.11%), despite differences in loss designs. Not surprisingly, Softmax Loss and Weighted Softmax Loss yield the same accuracies across all test datasets due to shared cross-entropy designs, becoming indistinguishable when trained on balanced data, per equations (2.8) and (2.9). Overall, all methods achieve results comparable to the softmax baseline and surpass the best published performance for MobileNetV2 trained on CIFAR-100, as shown in Table 4.9.

From Table 4.4, Balanced Softmax Loss outperforms other losses in terms of balanced test accuracy (Acc1: 57.96%) and especially tail classes (Acc1: 42.11%). 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. SEQL shows an overwhelming underperformance compared with the balanced training in table 4.3, and now appears severely limited. The difference in performance on tail classes compared to the softmax baseline shows that loss functions designed for imbalance can provide more nuanced performance gains.

Table 4.4: 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	0.5034	0.7336	0.7938	0.5385	0.2632
Balanced Softmax loss	**0.5796**	0.7650	0.8069	**0.6331**	**0.4211**
Equalization loss	0.0618	0.0647	0.0675	0.0592	0.0263
LDAM loss	0.4264	0.5899	0.6137	0.5444	0.2632

4.2.3 ViT-B/16

From Table 4.5, 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, as seen in table 4.9. Although ViTs has achieved state-of-the-art performance in image classification, their architecture differs from CNNs, as described in section 2.2.3, and may require careful fine-tuning for optimal results.

Table 4.5: 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	0.5432	0.5452	0.5427	0.5207	0.7105
Balanced Softmax loss	0.5628	0.5642	0.5640	0.5325	0.7105
Equalization loss	0.5582	0.5547	0.5462	0.5680	0.6842
LDAM loss	**0.5906**	**0.6013**	**0.5924**	**0.6095**	**0.7632**

Table 4.6: 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	0.1346	0.1865	0.2014	0.1361	0.0789
Balanced Softmax loss	**0.2460**	0.4244	0.4822	**0.2130**	0.0789
Equalization loss	0.1686	0.2778	0.3140	0.1361	**0.1053**
LDAM loss	0.1570	0.2750	0.3140	0.1361	0.0263

The results from Table 4.6 show low performance across all loss designs. Balanced Softmax Loss slightly improves results on balanced and middle categories, while Softmax Loss achieves the highest accuracy on the long-tailed dataset and head classes. The best performance on tail classes (Acc1: 10.53%) is achieved by SEQL, however, still unsatisfactory. 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 furhter adjustments in the training pipeline, as outlined in section 2.2.3.

4.2.4 ConvNeXt Base

From Table 4.7, CB Loss achieves the highest balanced test accuracy of 83.64% while the Softmax Loss, Weighted Softmax loss, and BS loss tie for top performance on the long-tailed dataset (Acc1: 94.74%). Softmax Loss and weighted softmax loss yield same perfromance across all categories due to their shared design on balanced datasets. The saturation of tail-class performance is likely due to the quality or quantity of tail classes. For comparison, the best published result for a ConvNext architecture trained with CIFAR-100 achieve an accuarcy of 94.04% (see table 4.9), the best performance of on the balanced test set is in alignment yields an accuracy of 83.98%, implying room for improvement in the training pipeline.

Table 4.7: 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	**0.8398**	0.8344	0.8389	0.7929	0.9211
Balanced Softmax loss	0.8364	0.8344	0.8365	0.7988	**0.9474**
Equalization loss	0.8322	0.8392	0.8412	0.8107	0.9211
LDAM loss	0.8316	0.8373	0.8412	0.8047	0.8947

Table 4.8 shows that Balanced Softmax Loss significantly outperforms other methods on the balanced test set (Acc1: 64.60%) and tail classes (Acc1: 57.89%). Softmax achieves the best performance on the long-tailed dataset, as well as on head and middle classes. Notably, SEQL delivers comparable results across all categories, in contrast to its performance with other model architectures, where it fell short compared to other loss designs.

4.3 Benchmarks

Table 4.9 compares the best results from the models trained on the balanced CIFAR-100 dataset in this thesis with published benchmarks using the same back-

Table 4.8: 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	0.3436	0.5756	0.6469	0.3254	0.1053
Balanced Softmax loss	**0.6460**	0.8230	0.8685	0.6509	**0.5789**
Equalization loss	0.5948	0.7973	0.8460	0.6272	0.4737
LDAM loss	0.3770	0.5956	0.6445	0.4260	0.2632

bone architectures. This comparison serves as a reference to ensure the results align with expectations and to identify any potential discrepancies.

For MobileNetV2, the results in this thesis surpass the published benchmarks. This could be attributed to improved loss function designs. On the other hand, both ResNet-50 and ConvNeXt-Base exhibits lower accuracies compared to benchmarks, which may be explained by differences in training pipelines.

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, as outlined in 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 could lead to an understanding of these discrepancies.

Table 4.9: Comparison of model performance (top-1 accuracy) with published benchmarks. The models are trained and tested on a balanced CIFAR-100 dataset. The performances are achieved with varying the techniques, and are meant as a reference and not a direct comparison.

Model	**Result (Acc1)**	**Benchmark Result (Acc1)**
ResNet-50	83.24%	86.90% [94]
MobileNetV2	80.78%	73.20% [95]
ViT-B/16	59.06%	93.51% [92]
ConvNeXt-Base	83.98%	94.04% [92]

4.4 Performance Comparisons

4.4.1 Model Performance

The models are evaluated based on the mean and standard deviation of their results across all loss functions. These statistics are visualized in Figure 4.1 and detailed in Table 4.10. Here, the performances of the models MobileNetV2, ResNet-

50, ViT-B/16, and ConvNeXt-Base are 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. 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.

Ignoring the sub-par performance of the ViT-B/16 architecture, the shorter errorbars for ConvNeXt-Base in figure 4.1 indicate that its architecture may be more robust to class-sensitive strategies than other architechtures. Additionally, this architecture displays the greatest mean accuracies across all categories. Contrary, the performance of MobileNetV2 shows a greater sensitivity to the choice of loss design with the exception of ResNet-50 on tail classes. 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. Overall, ConvNeXt-Base maintains strong and stable performance across all categories, benefiting from Balanced Softmax Loss as seen in table 4.8.

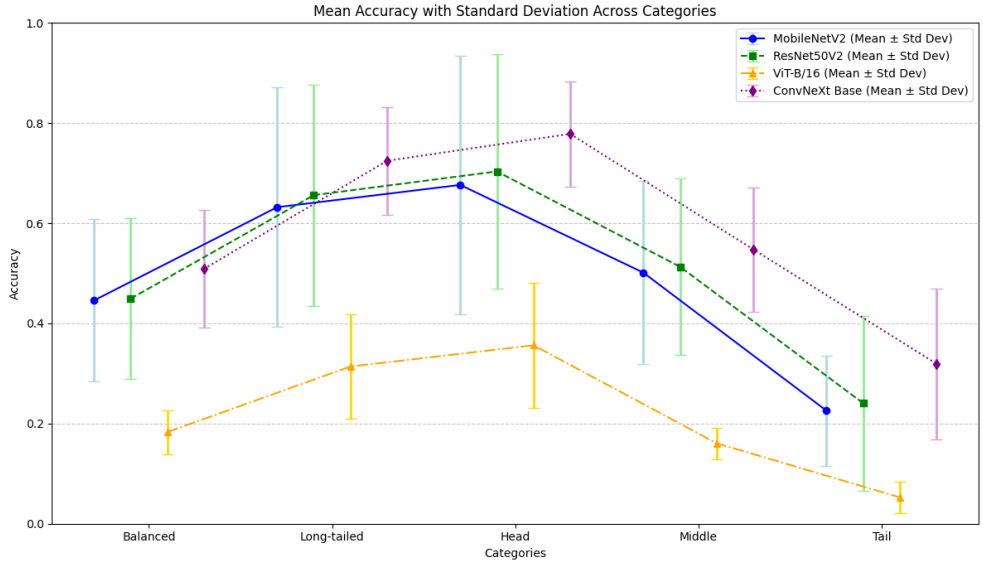


Figure 4.1: Mean Accuracy with Standard Deviation Across Categories for MobileNetV2, ResNet-50, ViT-B/16, and ConvNeXt Base trained with CIFAR-100-LT.

Table 4.10: Performance Summary Across Categories (Mean ± Std)

Model	Balanced	Long-tailed	Head	Middle	Tail
MobileNetV2	0.4459 ± 0.1623	0.6320 ± 0.2391	0.6765 ± 0.2585	0.5013 ± 0.1832	0.2256 ± 0.1107
ResNet-50	0.4494 ± 0.1605	0.6561 ± 0.2207	0.7035 ± 0.2348	0.5131 ± 0.1767	0.2406 ± 0.1743
ViT-B/16	0.1830 ± 0.0438	0.3139 ± 0.1047	0.3563 ± 0.1250	0.1606 ± 0.0315	0.0526 ± 0.0315
ConvNeXt-Base	0.5088 ± 0.1170	0.7247 ± 0.1077	0.7784 ± 0.1050	0.5477 ± 0.1243	0.3196 ± 0.1505

4.4.2 Loss Comparison

The Balanced Softmax Loss emerges as the overall best performing loss design across all models with the least variance in performance across test categories, excluding the performances of ViT-B/16, as figures 4.2, 4.3, and 4.5 illustrate. For reference, the performance of ViT-B/16 is illustrated in figure 4.4. From figure 4.5, the SEQL stands out in performance across all categories, and nearly competes with BS loss in terms of variance. In contrast, SEQL exhibits the worst performance on both ResNet-50 and MobileNetV2 (see Figures 4.2 and 4.3), possibly because these architectures are less robust to the large negative logits and masking strategy enforced by SEQL as opposed to ConvNeXt-Base.

These result are further underlined in figure 4.6, where the mean and standard deviation of the loss functions across the four backbone architectures are illustrated. Here, the Balanced Softmax Loss shows a better average perfomance on tail classes, while also displaying the smoothest variation across evaluation categories, ignoring the SEQL. Additionally, the standard deviation suggests that the perfomance of the Balanced Softmax Loss on tail classes depend on the model architecture. Table 4.11 display the exact values of the mean and standard deviations shown in figure 4.6.

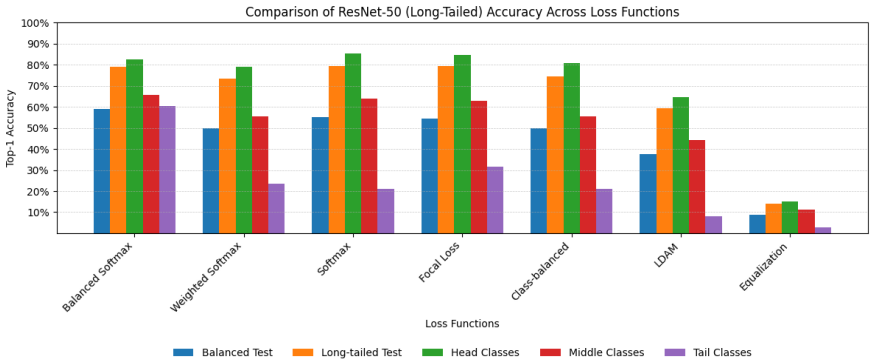


Figure 4.2: ResNet-50 top 1 accuracy across test categories (Balanced, Long-tailed, Head, Middle, Tail) for different loss functions.

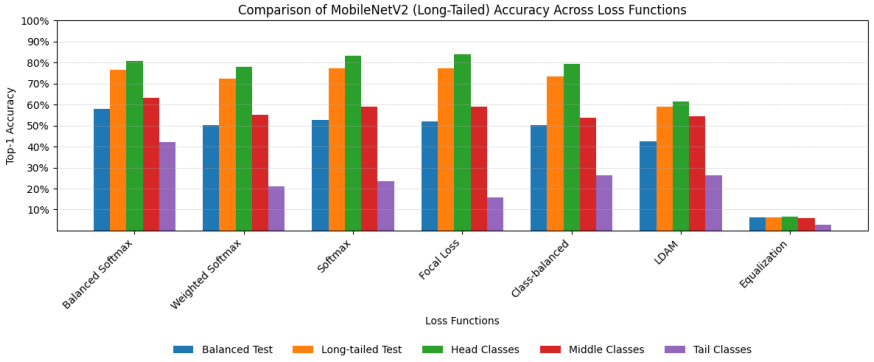


Figure 4.3: MobileNetV2 top 1 accuracy across test categories (Balanced, Long-tailed, Head, Middle, Tail) for different loss functions.

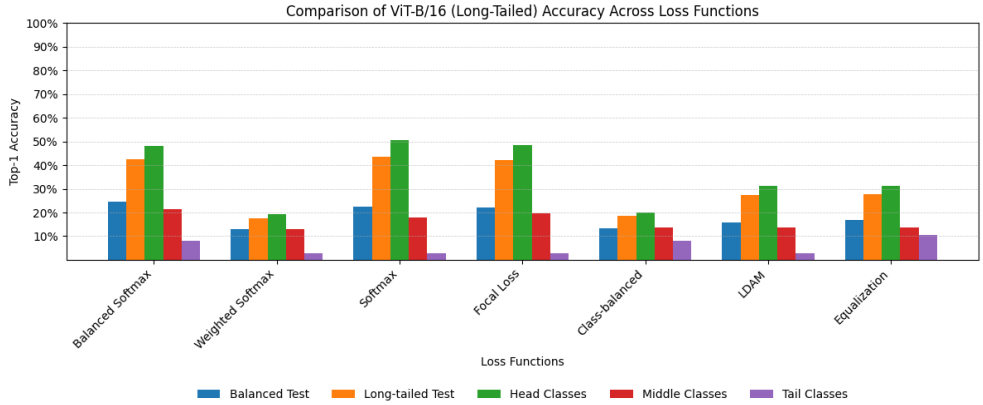


Figure 4.4: ViT-B/16 top 1 accuracy across test categories (Balanced, Long-tailed, Head, Middle, Tail) for different loss functions.

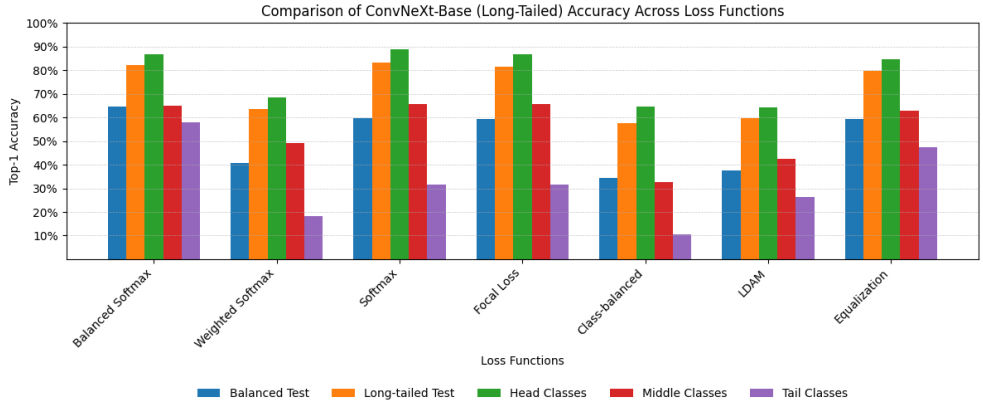


Figure 4.5: ConvNeXt-Base top 1 accuracy across test categories (Balanced, Long-tailed, Head, Middle, Tail) for different loss functions.

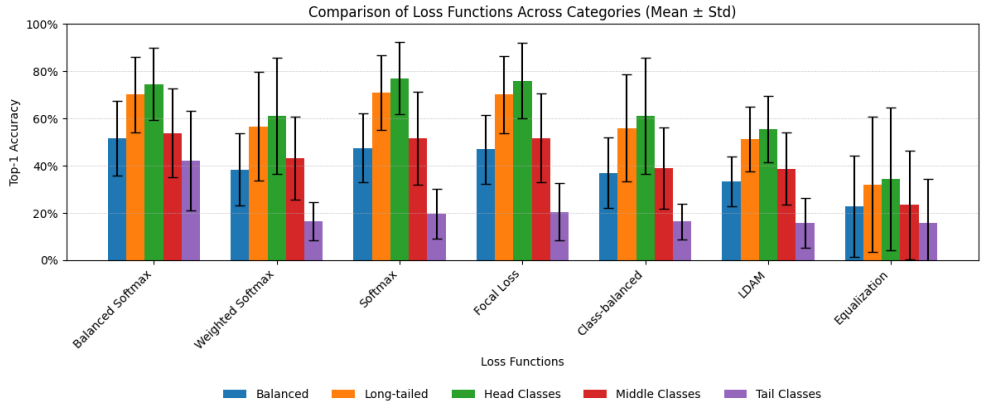


Figure 4.6: Performance trends of different loss functions across evaluation categories. Error bars indicate the standard deviation of accuracy across models, highlighting variability in performance for each loss function.

Table 4.11: 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	0.3842 ± 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
Class-balanced	0.3697 ± 0.1500	0.5602 ± 0.2259	0.6123 ± 0.2454	0.3891 ± 0.1720	0.1645 ± 0.0753
LDAM	0.3337 ± 0.1041	0.5136 ± 0.1377	0.5548 ± 0.1396	0.3876 ± 0.1520	0.1579 ± 0.1069
Equalization	0.2285 ± 0.2151	0.3199 ± 0.2860	0.3445 ± 0.3028	0.2337 ± 0.2289	0.1579 ± 0.1852

4.5 Summary and Discussion

Balanced Softmax and Tail-Class Performance By a large margin, the overall best performance on tail classes was achieved by the Balanced Softmax Loss, which proved to be consistent across all models while also achieving competitive results on other test categories. Balanced Softmax works by shifting logits to benefit minority classes, effectively countering the natural bias toward majority classes. However, as discussed below, this re-scaling can sometimes come at the expense of head-class performance.

Head vs. Tail Performance Trade-Off Although Balanced Softmax excels at improving tail-class accuracy, this improvement often comes with a trade-off in head-class performance. For example, while ResNet-50 achieved the highest top-1 accuracy of 60.53% on tail classes using Balanced Softmax (compared to 57.89% on ConvNeXt Base), its head-class accuracy (82.70%) lagged behind that of ConvNeXt Base (86.85%). This trade-off highlights the challenge of optimizing for both head and tail classes simultaneously, indicating that Balanced Softmax prioritizes boosting tail classes more aggressively. Future work could explore strategies to fine-tune or balance this trade-off.

SEQL In contrast, SEQL consistently underperformed across most models and test categories, with the exception of ConvNeXt-Base. A possible reason is that SEQL down-weights negative gradients, which can inadvertently weaken the model's ability to develop fine-grained distinctions among classes (see Section 2.4.6). The ConvNeXt architecture includes design elements such as LayerNorm (see Section 2.2.6) that might mitigate the adverse effects of heavy gradient masking. However, on small datasets like long-tailed CIFAR-100, down-weighting negative gradients deprives the model of critical examples for training, further impairing the learning of subtle class distinctions.

LDAM The LDAM loss could potentially be improved by combining it with a deferred re-weighting scheme (DRW), as introduced by Cao et al. [13]. With DRW, the model applies class-weighting only after a fixed number of epochs. Related work [23] demonstrates that DRW significantly boosts performance on long-tailed datasets, in contrast to using LDAM alone.

CNNs vs. ViTs CNN architectures (ResNet-50, MobileNetV2, and ConvNeXt Base) generally displayed better performance under the setup employed in this thesis. In contrast, ViT-B/16 underperformed significantly across all loss designs, categories, and training datasets, with its best accuracy reaching only 59.06% on a balanced training dataset, compared to a benchmark of 93.95% [92]. This discrepancy suggests that the default ViT-B/16 architecture may not be well-suited for smaller or long-tailed datasets without further adaptation. Transformers typically excel with large-scale, balanced data; hence, additional optimization techniques (e.g., specialized data augmentations, longer training, or advanced regularization) may be necessary to exploit ViT-B/16's full potential [23, 89].

Saturation on Balanced CIFAR-100 ResNet-50, MobileNetV2, and ConvNeXt-Base all displayed high, nearly equal tail-class performances when trained on the balanced version of CIFAR-100 (see Tables 4.1, 4.3, and 4.7). This suggests that tail-class accuracy may have reached a saturation point due to the inherently limited number of samples in these minority classes. Testing on larger, more diverse datasets would help clarify whether these accuracies can be further improved.

Optimizer Choice in Long-Tailed Learning Studies have shown that optimizer choice can influence long-tailed classification performance due to implicit biases [96, 23]. Specifically, SGD with momentum tends to correlate more strongly with class frequency, amplifying the advantage of head classes. In contrast, Adam exhibits a weaker correlation between class frequency and weight norms, potentially influencing re-weighting schemes [23].

Given these nuances, Adam was chosen primarily for its faster convergence and minimal hyperparameter tuning in this study. However, future work could explore switching to SGD to see if this yields better overall performance. For instance, a carefully tuned SGD might further improve tail-class accuracy when combined with a re-weighting strategy (like DRW).

Dataset Considerations The decision to derive test data directly from the training set, rather than creating a separate validation set with a matched long-tailed distribution (see Section 3.2), may have influenced the reported results.

Likewise, the combinations of model architectures and loss designs could benefit from training on larger long-tailed benchmarks like ImageNet-LT. These larger datasets follow a more natural Pareto distribution rather than an exponential decay, better mirroring real-world scenarios (see Sections 2.1 and 3.2). However, scaling to ImageNet-LT also introduces higher computational and resource demands.

Fine-Tuning and Transfer Learning Transfer learning from large-scale datasets (e.g., ImageNet) is widely acknowledged as a means of mitigating data scarcity in long-tailed problems [14, 92, 88]. In this study, all layers of the pretrained models were fine-tuned rather than freezing any subset of them.

Fine-tuning the entire network allows the model to fully adapt its learned features to the CIFAR-100 dataset, potentially benefiting minority classes that differ significantly from ImageNet classes [92].

However, this comprehensive fine-tuning introduces two notable trade-offs. First, updating all layers raises the risk of overfitting [14], especially when dealing with a smaller dataset such as CIFAR-100. Second, because the lower layers often capture general, transferable features (e.g., edges, shapes), fully retraining these layers can lead to a loss of these broad representations [14, 44]. In future work, partial freezing or layer-wise fine-tuning could be investigated to preserve the generality of early-stage features while still adapting higher layers to the characteristics of the long-tailed CIFAR-100 distribution.

Chapter 5

Conclusion and Future Work

This section revisits the goals outlined in Section 1.1.1, summarizes the main findings from the experiments, and suggests directions for future work.

5.1 Revisiting the Goals of the Thesis

In revisiting the goals of this thesis, it has become clear that the investigation of the deep long-tailed learning technique Class Re-Balancing has yielded valuable insight. The goals and findings are presented in the following.

Investigate the efficacy of long-tailed learning methods by assessing their performance on tail classes without sacrificing accuracy on head classes. The findings included that performance on tail classes often comes with a trade-off in performance on head classes. Similarly, adequate performances on all classes was often attributed to the performance on head classes and not inherently on tail classes. Nevertheless, Balanced Softmax Loss convincingly overcame this challenge across CNN-based architectures. This aligns with the goal of investigating the ability of deep long-tailed learning techniques to maintain overall accuracy.

Understand how model design affects the performance of deep long-tailed learning methods. The evaluations of different model architectures, namely the CNN-based and Vision Transformers, in combination with different class re-balancing techniques has revealed that model design has an effect on the performance of long-tailed learning. Specifically, ConvNeXt-Base showed robustness

Provide a comprehensive insight in comparisons that inform the choice of methods for long-tailed distributions. These findings contribute to a more comprehensive understanding of what to consider when training a deep network with long-tailed data, and, in doing so, serves as a guide to deep learning practitioners seeking more informed choices when encountered with a long-tailed problem.

5.2 Summary of Main Findings

The experiments conducted in this thesis underscore the complexity of deep long-tailed learning and highlight the importance of carefully combining model architectures, loss functions, and training configurations.

Among the evaluated loss designs, the Balanced Softmax Loss consistently provided the best performance on minority (tail) classes across multiple CNN backbones. By re-scaling logits according to class frequencies, Balanced Softmax effectively mitigates the natural bias toward majority (head) classes. Although achieving comparable performances on head classes, the results indicate that this improvement sometimes comes at the expense of head-class accuracy, reflecting a broader trade-off in optimizing both head and tail classes simultaneously.

Equalization Loss underperformed across all models with the exception of ConvNext-Base, indicating a strong correlation with model robustnes, dataset size, and down-weighting of gradients. Likewise, LDAM loss, tested without deferred re-weighting (DRW), did not match the consistency of Balanced Softmax, reaffirming that additional sampling strategies might be needed to fully realize LDAM's potential.

The ViT-B/16 architecture showed a pronounced performance gap compared not only to CNN-based architectures but also to its own reported benchmarks. Transformers often require large-scale and well-balanced datasets, as well as careful configuration of hyperparameters and training schemes, to perform at their peak. By contrast, CNN backbones generally demonstrated stronger performance and robustness under long-tailed conditions.
Overall, these results illustrate that improvements in tail-class accuracy often come with compromises in head-class performance and overall accuracy. This underscores the importance of carefully balancing strategies in long-tailed learning, including backbone architecture, loss function design, dataset size, parameter fine-tuning, and optimization strategies.

5.3 Future Work

Scaling to Larger Datasets Future investigation would benefit from evaluating experiments on large-scale, long-tailed benchmarks, such as ImageNet-LT or iNaturalist, to gain more insight into the performance of models and long-tailed techniques. The greater variety in tail classes and more realistic Pareto-like distributions could reveal whether the observed trends hold at scale, especially in the case of transformers, which typically require larger datasets and additional hyperparameter tuning. Training ViT-B/16 on a large-scale dataset with fine-tuning schedules tailored to preserve lower-layer feature representations may unlock its full potential and close the performance gap observed here.

Additionally, training ViT-B/16 on larger datasets while carefully considering its configurations could prove its full potential. Furthermore, evaluating the LDAM including the DRW and, potentially, combined with other loss functions on CIFAR-100-LT could reveal under which conditions tail-class performance is

maximized. Other techniques, such as re-sampling in combination with class re-balancing, could be explored. Finally, combining strong performances and ensemble learning can reveal extraordinary performances.

Refining Class-Sensitive Learning with Other Long-Tailed Methods Other studies have shown that combining LDAM with deferred re-weighting (DRW) on CIFAR-100-LT enchance performance on tail classes [13, 23]. Investigating how LDAM-DRW interacts with other re-balancing strategies or advanced optimizers could pinpoint conditions under which tail-class performance is maximized.

Freezing Model Parameters Given the trade-offs of fully fine-tuning pre-trained models, exploring partial freezing or gradual unfreezing of layers might better retain universal features while still adapting effectively to the distribution of minority classes. This layered approach could mitigate overfitting risks, particularly for smaller long-tailed datasets like CIFAR-100-LT.

Combining Approaches and Ensemble Learning In practice, a single technique may not suffice to handle highly skewed class distributions. Future work could explore combining the strongest methods, such as Balanced Softmax, and integrating them into ensemble architectures. Ensemble learning may help strike an optimal balance between tail-class gains and overall classification accuracy.

Bibliography

[1] 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.

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

[3] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV]. URL: https://arxiv.org/abs/1512.03385.

[4] 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.

[5] 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.

[6] Mateusz Buda, Atsuto Maki, and Maciej A. Mazurowski. "A systematic study of the class imbalance problem in convolutional neural networks". In: *Neural Networks* 106 (Oct. 2018), pp. 249–259. ISSN: 0893-6080. DOI: 10.1016/j.neunet.2018.07.011. URL: http://dx.doi.org/10.1016/j.neunet.2018.07.011.

[7] 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.

[8] 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.

[9] Yifan Zhang et al. *Deep Long-Tailed Learning: A Survey*. 2023. arXiv: 2110.04596 [cs.CV]. URL: https://arxiv.org/abs/2110.04596.

[10] Mark Sandler et al. *MobileNetV2: Inverted Residuals and Linear Bottlenecks*. 2019. arXiv: 1801.04381 [cs.CV]. URL: https://arxiv.org/abs/1801.04381.

[11] Zhuang Liu et al. *A ConvNet for the 2020s*. 2022. arXiv: 2201.03545 [cs.CV]. URL: https://arxiv.org/abs/2201.03545.

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

[13] 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.

[14] X. L. Chaitanya Asawa. *Cs231n: Convolutional neural networks for visual recognition*. Stanford, [Online]. Available: http://cs231n.github.io/.

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

[16] 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.

[17] 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.

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

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

[20] 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.

[21] Weiyang Liu et al. *Large-Margin Softmax Loss for Convolutional Neural Networks*. 2017. arXiv: 1612.02295 [stat.ML]. URL: https://arxiv.org/abs/1612.02295.

[22] Feng Wang et al. "Additive Margin Softmax for Face Verification". In: *IEEE Signal Processing Letters* 25.7 (July 2018), pp. 926–930. ISSN: 1558-2361. DOI: 10.1109/lsp.2018.2822810. URL: http://dx.doi.org/10.1109/LSP.2018.2822810.

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

[24] 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.

[25] 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.

[26] Connor Shorten and Taghi M. Khoshgoftaar. "A survey on Image Data Augmentation for Deep Learning". In: *Journal of Big Data* 6.1 (2019), p. 60. DOI: 10.1186/s40537-019-0197-0. URL: https://doi.org/10.1186/s40537-019-0197-0.

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

[28] 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.

[29] 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.

[30] 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.

[31] 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.

[32] 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.

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

[34] 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.

[35] 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.

[36] 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)* (2023), pp. 1–6. URL: https://api.semanticscholar.org/CorpusID:266486570.

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

[38] 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.

[39] Yann LeCun et al. "Comparison of learning algorithms for handwritten digit recognition". In: *International Conference on Artificial Neural Networks* (Jan. 1995).

[40] Yann 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.

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

[42] mathworks.com. *What Is a Convolutional Neural Network?* Available: https://www.mathworks.com/discovery/convolutional-neural-network.html. Accessed: 28-11-2024, 2024.

[43] Vinod Nair and Geoffrey E. Hinton. "Rectified linear units improve restricted boltzmann machines". In: *Proceedings of the 27th International Conference on International Conference on Machine Learning*. ICML'10. Omnipress, 2010, pp. 807–814.

[44] Jason Yosinski et al. *How transferable are features in deep neural networks?* 2014. arXiv: 1411.1792 [cs.LG]. URL: https://arxiv.org/abs/1411.1792.

[45] ibm.com. *What are cconvolutional neural networks?* Available: https://www.ibm.com/think/topics/convolutional-neural-networks. Accessed: 26-12-2024, 2024.

[46] wandb.ai. *Understanding Logits, Sigmoid, Softmax, and Cross-Entropy Loss in Deep Learning*. Available at: https://wandb.ai/amanarora/Written-Reports/reports/Understanding-Logits-Sigmoid-Softmax-and-Cross-Entropy-Loss-in-Deep-Learning--Vmlldzo0NDMzNTU3. Accessed: 31-12-2024, 2022.

[47] medium.com. *What is the Inductive Bias in Machine Learning?* Available at: https://medium.com/@chkim345/what-is-the-inductive-bias-in-machine-learning-212a5f53e9aa. Accessed: 31-12-2024, 2020.

[48] Olga Russakovsky et al. "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision (IJCV)* 115.3 (2015), pp. 211–252. DOI: 10.1007/s11263-015-0816-y.

[49] 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.

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

[51] Ashish Vaswani et al. *Attention Is All You Need*. 2023. arXiv: 1706.03762 [cs.CL]. URL: https://arxiv.org/abs/1706.03762.

[52] v7labs.com. *Vision Transformer Guide*. https://www.v7labs.com/blog/vision-transformer-guide. Accessed: 18-12-2024. 2024.

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

[54] TorchVision maintainers and contributors. *TorchVision: PyTorch's Computer Vision library*. Available at: https://github.com/pytorch/vision. Accessed: 18-12-2024. 2016.

[55] medium.com. *The Annotated ResNet-50*. Available: https://towardsdatascience.com/the-annotated-resnet-50-a6c536034758. Accessed: 03-01-20245, 2022.

[56] 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.

[57] Yunhui Guo et al. *Depthwise Convolution is All You Need for Learning Multiple Visual Domains*. 2019. arXiv: 1902.00927 [cs.CV]. URL: https://arxiv.org/abs/1902.00927.

[58] kungfu.ai. *ConvNeXt: A Transformer-Inspired CNN Architecture*. Available: https://www.kungfu.ai/blog-post/convnext-a-transformer-inspired-cnn-architecture. Accessed: 28-11-2024, 2023.

[59] Saining Xie et al. *Aggregated Residual Transformations for Deep Neural Networks*. 2017. arXiv: 1611.05431 [cs.CV]. URL: https://arxiv.org/abs/1611.05431.

[60] Dan Hendrycks and Kevin Gimpel. *Gaussian Error Linear Units (GELUs)*. 2023. arXiv: 1606.08415 [cs.LG]. URL: https://arxiv.org/abs/1606.08415.

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

[62] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG]. URL: https://arxiv.org/abs/1412.6980.

[63] pytorch.org. *CrossEntropyLoss*. Available at: https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html. Accessed: 20-11-2024. 2024.

[64] pytorch.org. *CIFAR100*. Available at: https://pytorch.org/vision/0.17/generated/torchvision.datasets.CIFAR100.html. Accessed: 13-12-2024. 2024.

[65] Sinno Jialin Pan and Qiang Yang. "A Survey on Transfer Learning". In: *IEEE Transactions on Knowledge and Data Engineering* 22.10 (2010), pp. 1345–1359. DOI: 10.1109/TKDE.2009.191.

[66] G. E. Hinton and R. R. Salakhutdinov. "Reducing the Dimensionality of Data with Neural Networks". In: *Science* 313.5786 (2006), pp. 504–507. DOI: 10.1126/science.1127647.

[67] Yang Fu, Xiangnian Huang, and Yunfeng Li. "Horse Breed Classification Based on Transfer Learning". In: *Proceedings of the 4th International Conference on Advances in Image Processing*. Association for Computing Machinery, 2021, pp. 42–47. DOI: 10.1145/3441250.3441264.

[68] pytorch.org. *MobileNetV2*. Accessed: 22-12-2024. 2024. URL: https://pytorch.org/vision/main/models/generated/torchvision.models.mobilenet_v2.html#torchvision.models.mobilenet_v2.

[69] pytorch.org. *resnet50*. Available at: https://pytorch.org/vision/main/models/generated/torchvision.models.resnet50.html. Accessed: 26-11-2024. 2024.

[70] pytorch.org. *convnext base*. Available at: https://pytorch.org/vision/main/models/generated/torchvision.models.convnext_base.html. Accessed: 26-11-2024. 2024.

[71] Yijin Huang et al. *Identifying the key components in ResNet-50 for diabetic retinopathy grading from fundus images: a systematic investigation*. 2022. arXiv: 2110.14160 [eess.IV]. URL: https://arxiv.org/abs/2110.14160.

[72] Gizeaddis Lamesgin Simegn, Mizanu Zelalem Degu, and Geletaw Sahle Tegenaw. "Cervical Cancer Histopathological Image Classification Using Imbalanced Domain Learning". In: *Advancement of Science and Technology*. Springer Nature Switzerland, 2025, pp. 3–20.

[73] Benedicta Nana Esi Nyarko et al. "Comparative Analysis of AlexNet, Resnet-50, and Inception-V3 Models on Masked Face Recognition". In: *2022 IEEE World AI IoT Congress (AIIoT)*. 2022, pp. 337–343. DOI: 10.1109/AIIoT54504.2022.9817327.

[74] Ekin D. Cubuk et al. *RandAugment: Practical automated data augmentation with a reduced search space*. 2019. arXiv: 1909.13719 [cs.CV]. URL: https://arxiv.org/abs/1909.13719.

[75] Le Liu et al. "A Transfer Learning Method Based on ResNet Model". In: *Data Mining and Big Data*. Springer Singapore, 2021, pp. 250–260.

[76] Mohammad Razavi, Samira Mavaddati, and Hamidreza Koohi. "ResNet deep models and transfer learning technique for classification and quality detection of rice cultivars". In: *Expert Systems with Applications* 247 (2024), p. 123276. DOI: https://doi.org/10.1016/j.eswa.2024.123276.

[77] medium.com. *A Guide to Transfer Learning with Keras Using ResNet50*. Available at: https://medium.com/@kenneth.ca95/a-guide-to-transfer-learning-with-keras-using-resnet50-a81a4a28084b. Accessed: 23-12-2024, 2019.

[78] Muhammad Shafiq and Zhaoquan Gu. "Deep Residual Learning for Image Recognition: A Survey". In: *Applied Sciences* 12.18 (2022). DOI: 10.3390/app12188972.

[79] Piyush Nagpal, Shivani Atul Bhinge, and Ajitkumar Shitole. "A Comparative Analysis of ResNet Architectures". In: *2022 International Conference on Smart Generation Computing, Communication and Networking (SMART GENCON)*. 2022, pp. 1–8. DOI: 10.1109/SMARTGENCON56628.2022.10083966.

[80] Aaditya Surya et al. *Enhanced Breast Cancer Tumor Classification using MobileNetV2: A Detailed Exploration on Image Intensity, Error Mitigation, and Streamlit-driven Real-time Deployment*. 2024. arXiv: 2312.03020 [eess.IV]. URL: https://arxiv.org/abs/2312.03020.

[81] Umamaheswari S et al. "Performance Analysis of ResNet50 Architecture based Pest Detection System". In: *2023 9th International Conference on Advanced Computing and Communication Systems (ICACCS)*. Vol. 1. 2023, pp. 578–583. DOI: 10.1109/ICACCS57279.2023.10112802.

[82] Tika B Shahi et al. "Fruit classification using attention-based MobileNetV2 for industrial applications". In: *PLOS ONE* 17.2 (2022), e0264586. DOI: 10.1371/journal.pone.0264586.

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

[84] Alexander Kolesnikov et al. *Big Transfer (BiT): General Visual Representation Learning*. 2020. arXiv: 1912.11370 [cs.CV]. URL: https://arxiv.org/abs/1912.11370.

[85] A. A. Asiri et al. "Advancing Brain Tumor Classification through Fine-Tuned Vision Transformers: A Comparative Study of Pre-Trained Models". In: *Sensors (Basel, Switzerland)* 23.18 (2023), p. 7913. DOI: 10.3390/s23187913.

[86] pytorch.org. *convnextbase*. Available at: https://pytorch.org/vision/main/models/generated/torchvision.models.convnext_base.html. Accessed: 04-12-2024. 2024.

[87] pytorch.org. *StepLR*. Available at: https://pytorch.org/docs/stable/generated/torch.optim.lr_scheduler.StepLR.html. Accessed: 04-12-2024. 2024.

[88] Ibrahem Kandel and Mauro Castelli. "How Deeply to Fine-Tune a Convolutional Neural Network: A Case Study Using a Histopathology Dataset". In: *Applied Sciences* 10.10 (2020). DOI: 10.3390/app10103359.

[89] Ilya Loshchilov and Frank Hutter. *Fixing Weight Decay Regularization in Adam*. 2018. URL: https://openreview.net/forum?id=rk6qdGgCZ.

[90] A. L. Cauchy. *Leçons sur le calcul différentiel*. Paris, 1829.

[91] pytorch.org. *Adam*. Available at: https://pytorch.org/docs/stable/generated/torch.optim.Adam.html. Accessed: 04-12-2024. 2024.

[92] Peng Ye et al. *Partial Fine-Tuning: A Successor to Full Fine-Tuning for Vision Transformers*. 2023. arXiv: 2312.15681 [cs.CV]. URL: https://arxiv.org/abs/2312.15681.

[93] Juri Opitz. "A Closer Look at Classification Evaluation Metrics and a Critical Reflection of Common Evaluation Practice". In: *Transactions of the Association for Computational Linguistics* 12 (June 2024), pp. 820–836. DOI: 10.1162/tacl_a_00675.

[94] 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.

[95] 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.

[96] Daniel Soudry et al. *The Implicit Bias of Gradient Descent on Separable Data*. 2024. arXiv: 1710.10345 [stat.ML]. URL: https://arxiv.org/abs/1710.10345.

Appendix A

Experimental Setup Details

A.1 Data Preprocessing

Table A.1 summarizes the preprocessing steps applied to the training, validation, and test datasets.

Table A.1: 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]

A.2 Model Architecture Settings

The specifications of the model architectures can be seen in table A.2.

A.3 Hardware and Software Configurations

Tables A.3 and A.4 lists the hardware and software specifactions for the experiments in this thesis, respectively.

Table A.2: Model Architecture Settings

Model Name	Pretrained Weights	Modifications for CIFAR-100
ResNet-50 [69]	`ResNet50_Weights.IMAGENET1K_V2`	Replaced the `fc` layer with a 100-class fully connected layer
MobileNetV2 [68]	`MobileNet_V2_Weights.IMAGENET1K_V1`	Replaced classification layer with a 100-class fully connected layer
ViT-B/16 [4]	`timm vit_base_patch16_224` pretrained	Replaced the `head` with a 100-class fully connected layer
ConvNeXt-Base [70]	`ConvNeXt_Base_Weights.DEFAULT`	Replaced the final layer of the classifier with a 100-class fully connected layer

Table A.3: Hardware Specifications

Component	Specification
GPUs	4 NVIDIA TITAN X (Pascal), 12 GB each
RAM	125 GiB
Swap Space	63 GiB
CUDA Version	12.4
Driver Version	550.90.07

Table A.4: Software Specifications

Component	Specification
Operating System	Ubuntu 22.04.4 LTS (Jammy Jellyfish)
Python Version	3.11.8
Libraries	PyTorch ($\geq$ 1.7) Torchvision ($\geq$ 0.8.0) Timm (1.0.11) Tensorboard ($\geq$ 1.14)

A.4 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.
- **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.

Appendix B

Declaration of GAI

B.1 Declaration for the use of GAI

Below is the filled out declaration for the use of Generative Artificial Intelligence from Aarhus University.

Declaration for the use of Generative Artificial Intelligence in the project

Use of Generative Artificial Intelligence (GAI) in projects

In accordance with the new guidelines at Aarhus University on the use of generative artificial intelligence (GAI), you must include a declaration as an appendix if you have used GAI (such as ChatGPT, Microsoft Copilot (Bing), Google Gemini, custom GPTs etc..) in working on your project.

Exam assignments are assessed based on the learning objectives in the course description, and therefore the use of GAI will not in itself affect the assessment, provided that the rules for using GAI are observed.

Please remember to always check the rules for using GAI on studerende.au.dk.

How to fill out the declaration

In your declaration of use of GAI, you must as a minimum:

1. Declare that you have used GAI.
2. Specify the technology used, specifying the version number (ex ChatGPT 3.5 or Llama 2).
3. Describe how information has been generated, describe the inputs used, and explain how the output was used in your project report – see instructions on Studypedia
4. In agreement with your supervisor, you may also want to enclose a more detailed appendix with prompts and the outputs generated.

Examples of using GAI

Below are a number of examples of how you may have GAI tools in your project. You may specify additional ways that you have used GAI than those listed.
In the relevant cases, you must indicate which essential tools have been used and briefly describe how and with what benefit you have used them.

- To search or structure information
- For programming tasks
- For data analysis
- To produce figures
- To format text/formulas in your project
- To get feedback/improvements/proofreading of text and wordings
- To generate text used directly in the task (important, see below)
    - If you have rewritten texts generated by GAI but not directly quoted them, declare how you have used GAI.
    - If you have quoted directly from text generated by GAI, it must be quoted as a citation with a reference to the GAI tool you have used – see instructions here: AU Library.

It is suggested that the following form is used as a declaration (and please fill in the form in collaboration with your supervisor):

Declaration for the use of Generative Artificial Intelligence in the project

Name: Christine Midtgaard

Study number: 201404750

Project type: Master's Thesis

Project title: Deep Learning Techniques for Long-Tailed Image Classification: A Comparative Study of Loss Designs and Model Architectures

✔ · **I have used GAI to produce this project** *(tick)*

Describe which GAI tools you have used (remember version):

ChatGPT 4o

I have used GAI in the following way:

Some possible uses are listed as inspiration – delete or add as needed. For each relevant area, explain how GAI is used. For example, briefly describe how the information was generated, as well as explain how the output was used in your assignment.

- *to search or structure information*
- *for programming tasks*
- *for data analysis*
- *to produce figures*
- *to format text/formulas in your project*
- *to get feedback/improvements/proofreading of text and wording*
- *to generate text applied directly to the task*
- *other*

I have used ChatGPT to find synonyms and to enhance clarity and improve formulation in text that I have already written. Afterwards, I have reviewed and edited the content as needed to ensure its accuracy. I take full responsibility for the content of this thesis.