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, where a few classes dominate with abundant samples while many classes have sparse representation, pose significant challenges for traditional training methods. These imbalances often lead to models that perform well on majority classes but struggle to recognize or generalize to minority (tail) classes.

Acknowledgements

Contents

Chapter 1

Introduction

TODO: Emphasize head and tail classes.

Image classification is one of the main challenges computer vision.

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

This thesis focuses on the problem with long-tailed datasets. The problem with training a deep learning model on long-tailed datasets is that the model will effectively the data from the classes with most samples, and not the classes with few samples. The finsihed model will then not recognize an input from the tail classes. Most real-world datasets follows a long-tailed structure, hence the need for a reliable method to detect examples of tail-class data. The aim of this thesis is to try out some of the methods tackling the long-tailed problem for deep learning described in the paper *Deep Long-Tailed Learning: A Survey* by Zhang et al.[5] to find a method for long-tailed learning that works on a specific long-tailed dataset of images of moths taken around equator. The goal of the moth dataset is to identify species.

1.1 Problem Definition

TODO: Something about head and tail classes.

The goal of this project is to investigate deep learning models, like Convolutional Neural Networks (CNNs) and Vision Transformers (ViTs), and methods, mainly class re-balancing through cost-sensitive learning, and analyze their performance on datasets representaing a long-tailed structure.

Four models are investigated, where three of them are CNNs, and one is a ViT. The CNNs investigated are the MobileNetV2, ResNet50v2, and ConvNeXt Base,

while the ViT investigated is the ViT-B/16. All four models are pretrained on ImageNet, and further trained on both a balanced version of CIFAR100 and an long-tailed version of CIFAR100, called CIFAR100-LT, and introducing the long-tailed technique as class re-balancing in the loss functions by using re-weighting. All models are trained with Softmax Cross-Entropy Loss, Weighted Softmax Cross-Entropy Loss, Focal Loss, Class-Balanced Loss, Balanced Softmax Loss, LDAM Loss, and Equalization Loss. The main purpose of this master's thesis is to compare class re-balancing methods on a well respresenting type of models used in deep learning image recognition tasks. Furhter comparison with state-of-the-art methods for deep long-tailed learning will be made. The motivation for this comparison is to find a way to train on a real-world long-tailed dataset of moths to correctly identify species.

1.1.1 Goals of this thesis

The goals of this thesis are to:

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

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

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

1.1.2 Hypothesis

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

1.1.3 Approach

TODO: finish this section. This master's thesis consists of six steps, described below:

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

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

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

Fourth step is hyperparameter settings.

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

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

1.1.4 Scope of this thesis

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

1.2 Related Work

The challenge of long-tailed datasets has been extensively studied in the literature [5, 6].
TODO: Add references to re-weighting papers.

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) [7, 8]. ROS involves duplicating random samples from the minority classes, whereas RUS involves removing samples from the majority classes [5, 8]. 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 [5]. Another re-sampling technique, called Synthetic Minority Over-Sampling Technique (SMOTE) [7], involves synthetical generation of instances of the minority classes based on the existing data.

Transfer Learning Transfer learning techniques adress the issue of class imblance by transferring learned features from head classes to tail classes. Examples

include transferring the intra-class variance [9] and transferring semantic deep features [4].

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 [10, 11]. Common augmentation methods include Mixup [12], which combines pairs of samples to create interpolated examples; CutMix [13], which replaces regions of one image with patches from another; and UniMix [14], which focuses on calibrating the feature space for long-tailed distributions. Rare-class Sample Generator (RSG) [15] focuses on enhancing tail-class performance by transferring knowledge from head classes.

Decoupled Training Decoupled training [16] separates representation learning and classifier training. First, the model is trained to extract features from the entire dataset, afterwards, a re-balancing technique is applied during classification. Despite the simple framework, decoupling showed state-of-the art performance on long-tailed benchmarks.

Ensemple Learning Ensemble learning is a technique that combines multiple network modules (i.e. experts) to solve long-tailed problems. By doing so, the predictions of the different models can be combined, ultimately outperforming a single model [17, 18].

TODO: Margin Loss

> Margin loss. The hinge loss is often used to obtain a "max-margin" classifier, most notably in SVMs [Suykens and Vandewalle, 1999]. Recently, Large-Margin Softmax [Liu et al., 2016], Angular Softmax [Liu et al., 2017a], and Additive Margin Softmax [Wang et al., 2018a] have been proposed to minimize intra-class variation in predictions and enlarge the inter-class margin by incorporating the idea of angular margin. In contrast to the class-independent margins in these papers, our approach encourages bigger margins for minority classes. Uneven margins for imbalanced datasets are also proposed and studied in [Li et al., 2002] and the recent work [Khan et al., 2019, Li et al., 2019]. Our theory put this idea on a more theoretical footing by providing a concrete formula for the desired margins of the classes alongside good empirical progress. [19]

1.3 Reading Guide

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

Chapter 2

Background

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

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

2.1 Long-Tailed Datasets

Long-tailed datasets pose significant challenges in deep learning, as they represent an extreme form of class imbalance. Addressing these challenges is central to this thesis, which explores methods to improve model performance on underrepresented classes. This section outlines the structure of long-tailed distributions and their implications.

A balanced dataset is one where all classes are evenly represented, whereas imbalanced datasets feature varying sample sizes across classes. Long-tailed datasets are characterized by a significant class imbalance, where a few dominant classes account for most samples (head classes), while the majority of classes are underrepresented (tail classes) as depicted in Figure 2.1. This distribution is common for real-world datasets [20, 4]. For example, the iNaturalist, a popular benchmark for image classification, exhibits a long-tailed distribution of species [21]. Other benchmarks are constructed by sampling from datasets such as ImageNet [22] by using a Pareto distribution, which simulates long-tailed class distributions with a power-law decay [5, 23, 19].

CIFAR100-LT [19], derived from the CIFAR-100 dataset [24], serves as the primary dataset for the experiments conducted in this thesis. CIFAR-100 is a widely used benchmark in classification research due to its diverse class representation and manageable size. It consists of 60,000 32x32 color images divided into 100 classes, each with 600 samples. These are further split into 500 training images and 100 testing images per class. CIFAR100-LT is created by reducing the number of samples in certain classes of CIFAR-100 following an exponential decay en sample sizes, given by:

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

Where n_i is the number of samples in class i, n_{max} is the number of samples in the most frequent class, IR is the imbalance ratio, and C is the total number of classes [25].

Other long-tailed datasets follow a Pareto distribution with number of samplers per class as followed [4]:

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

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

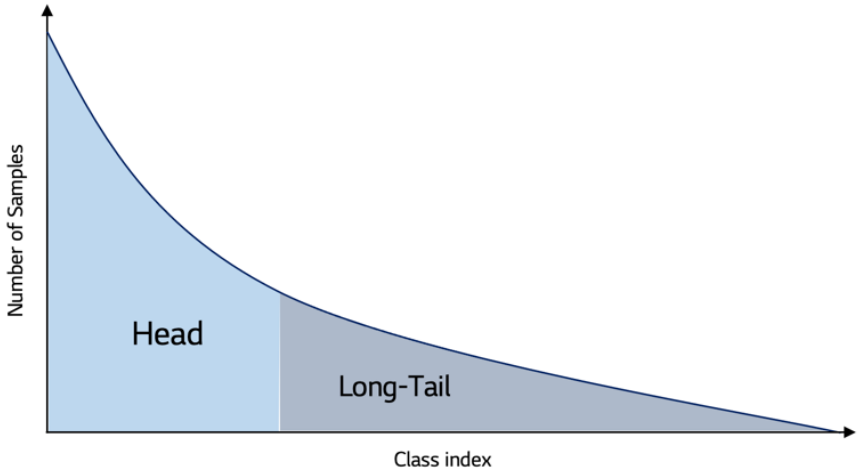

Figure 2.1: Illustration of a long-tailed distribution. Figure from [26].

Class imbalance has a profound impact on model performance compared to evenly distributed datasets [2, 27]. Deep networks trained on long-tailed datasets often exhibit biased performance, favoring head classes while performing poorly on tail classes [5]. Zhang et al. (2023) provide a comprehensive survey of methods addressing this challenge, categorizing current approaches into three main groups: class re-balancing, information augmentation, and module improvement. These methods will be further explored in section 2.3.

TODO: Write something of the following: "Let $\{x_i, y_i\}_{i=1}^{n}$ be the long-tailed training set, where each sample x_i has a corresponding class label y_i. The total number of training samples over K classes is $n = \sum_{k=1}^{K} n_k$, where n_k denotes 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. Without loss of generality, a common assumption in long-tailed learning [**31**, **32**] is that the classes are sorted by cardinality in decreasing order (i.e., if $i_1 < i_2$, then $n_{i_1} \geq n_{i_2}$, and $n_1 \gg n_K$). The imbalance ratio is then defined as $\frac{n_1}{n_K}$." [5]

2.2 Model Architecures

Deep learning has revolutionized image classification by introducing models capable of learning complex patterns and representations from data. Among these, Convolutional Neural Networks (CNNs) and Vision Transformers (ViTs) are chosen as the primary architectures used in this thesis due to their performance on image classification tasks. This section provides a theoretical foundation for these models, focusing on the specific architectures utilized: MobileNetV2 [28], ResNet50V2 [29], and ConvNeXt Base [30] as the CNN architectures, and ViT-B/16 [31] as the ViT architecture. These models were chosen to represent a spectrum of design paradigms; from the traditional ResNet to lightweight architectures and 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 Convolutional Neural Networks (CNNs) and, more recently, Vision Transformers (ViTs), the standard approach for image classification involved flattening a two-dimensional image matrix into a one-dimensional array and passing it through a Multilayer Perceptron (MLP), also known as a feed-forward neural network. MLPs are fully connected neural networks composed of an input layer, output layer, and one or more hidden layers. Being fully connected means that each neuron in a given layer is connected to all neurons in the next layer, forming a dense network. These connections are associated with weights and biases, which the network learns during training. Input features are fed into the input layer, propagated through hidden layers that add complexity to model nonlinear relationships, and yield predictions in the output layer. Known as universal approximators, MLPs can approximate any continuous function given sufficient neurons in the hidden layers [32, 33].

To illustrate the structure of a neural network, figure 2.2 shows an example of a feed-forward neural network with three input neurons, two hidden layers, each with four neurons, and two output neurons. This architecture could be used, for instance, to classify images of cats and dogs based on three input features, such as height, weight, and width of the animals. The input propagates through the network, with each neuron computing a weighted sum of its inputs followed by an optional nonlinearity. The final output is a prediction, where the class corresponding to the neuron with the highest value is chosen.

However, this simple neural network becomes insufficient for more complex problems, such as image classification, as it requires an increasing number of parameters. For instance, a 224×224 RGB image flattened is very large, making MLPs parameter-heavy and inefficient. The limitations of MLPs were addressed by Convolutional Neural Networks (CNNs), which introduced convolutional and pooling layers to effectively preserve and utilize the spatial information of pixels in two-dimensional images [32].

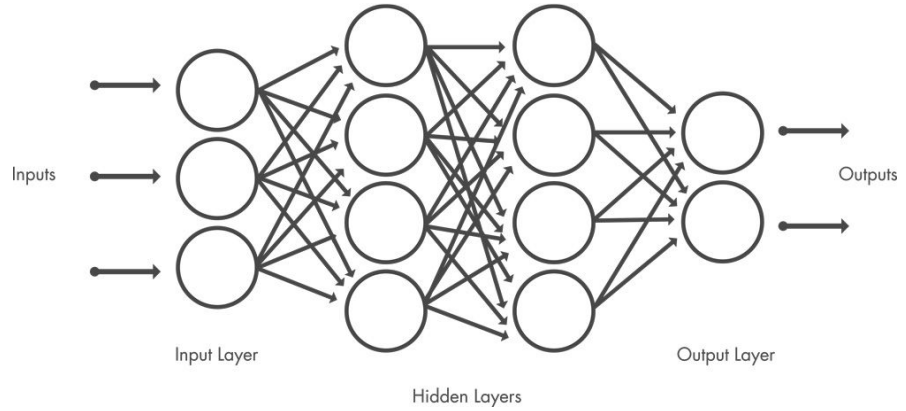

Figure 2.2: Layers of a neural network. Figure from [34]. TODO: Make this figure.

2.2.2 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) [35] 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 [36, 37, 32].

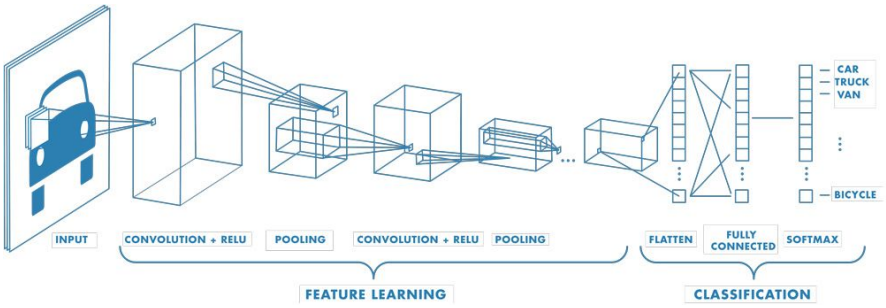

Figure 2.3: Illustration of a convolutional neural network. Figure from [34]. TODO: Make this figure.

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 layer [38]. 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 fully connected layer. Activation functions introduce nonlinearity, allowing the network to capture complex patterns. At the final stage, fully connected or global pooling layers aggregate the extracted features into predictions, enabling tasks such as classification or segmentation. CNNs use three-dimensional data for image classification: *width height*, *depth*. As an example, an image from the CIFAR-100 dataset $32 \times 32 \times 3$, 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 [38]. 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 TODO: make this figure. 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 TODO: reference. After each convolution, the ReLu transformation is applied to introduce non-linearity [38].

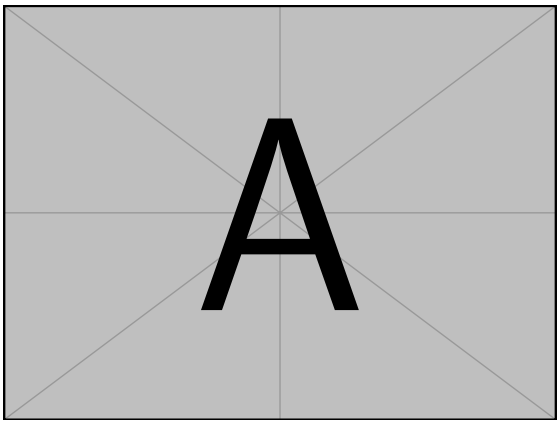

Figure 2.4: Illustration of a convolution. TODO: Make this figure.

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 [38]. Moreover, this contributes to making the detection of features invariant to scale and orientation.

Fully Connected (FC) 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 [38]. The raw output of the FC layer are called logits [39], 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 [40].

CNNs gained popularity after the introduction of LeNet-5 by LeCun et al. in 1998 [36], which showcased their capability to recognize handwritten digits.

The field advanced significantly in 2012 when AlexNet [37] won the ImageNet Challenge, 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 [41], GoogLeNet [42], and the state-of-the-art ResNet [29]. The CNN architectures explored in this thesis are detailed below.

ResNet50 Architecture

ResNet50 is a variant of the Residual Network (ResNet) architecture, developed by Microsoft Research (He et al.) in 2015 [29]. 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 [29]. 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 2.5, 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 architecure 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 $\mathcal{H}(x)$ [29].

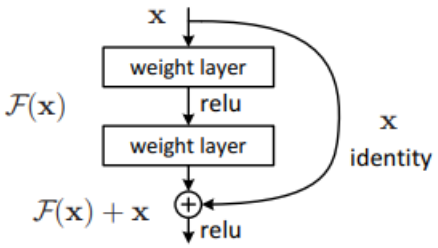

Figure 2. Residual learning: a building block.

Figure 2.5: Residual Learning. Figure from [29]. TODO: Make this figure.

The residual network is formed by stacking multiple layers of residual blocks, as the one depicted in figure 2.5. The ResNet architecture can have a varying number of layers, and, as the name implies, the ResNet50 variant has 50 layers. Other architecures include ResNet-34, ResNet-101, and ResNet-152 [43].

The ResNet50 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[44], as seen in figure 2.6.

MobileNetV2 Architecture

MobileNetV2 introduced by Sandler et al. [28] is a lightweight CNN model designed primarily to balance model accuracy and computational efficiency, making

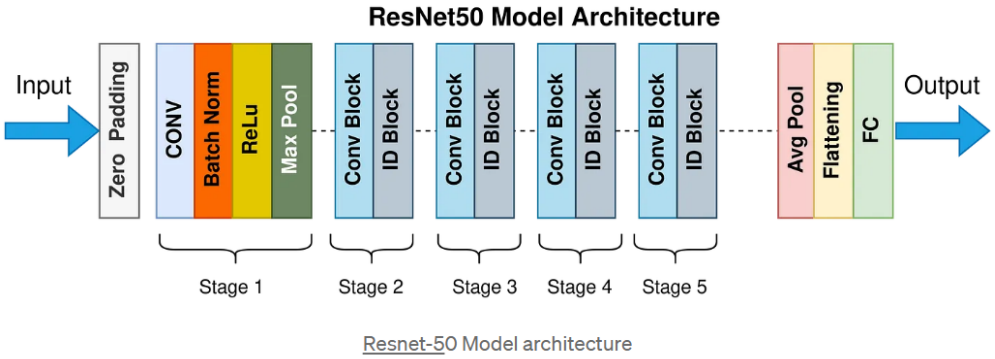

Figure 2.6: ResNet50 Architecture. Figure from `https://towardsdatascience.com/the-annotated-resnet-50-a6c536034758`. TODO: Make this figure.

it suitable for mobile or embedded devices. Building upon the original concepts of MobileNetV1 [45], 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 described above allow for identity mapping and improved gradient flow, MobileNetV2 employs an inverted residual structure [28]. 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 2.7, helps preserve crucial information and maintain a rich feature space without substantially increasing computational cost. The use of a linear bottleneck (i.e., no nonlinear activation in the low-dimensional projection) also helps prevent the destruction of useful information, further improving efficiency and accuracy.

Depthwise Separable Convolution Following MobileNetV1, MobileNetV2 relies on depthwise separable convolutions to factorize the convolution operation into two simpler operations [45]: depthwise convolution and pointwise convolution as shown in figure 2.8. The depthwise convolution applies a single filter to each input channel, and the pointwise convolution (a 1×1 convolution) then recombines the channels to produce the desired output. In comparison, a standard convolution both filtes and combines inputs into a new set of outputs. This approach reduces the parameter count and computational load, making the model suitable for devices with limited resources [45, 28].

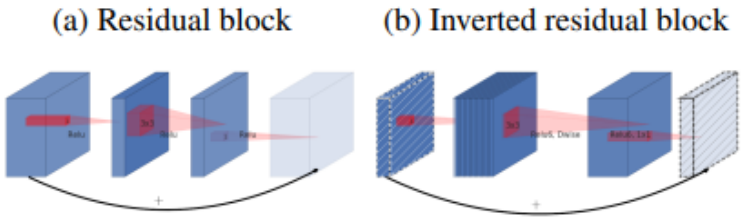

Figure 3: The difference between residual block [8, 30] and inverted residual. Diagonally hatched layers do not use non-linearities. We use thickness of each block to indicate its relative number of channels. Note how classical residuals connects the layers with high number of channels, whereas the inverted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

Figure 2.7: Residual and Inverted Residual Blocks. Figure from [28]. TODO: Make this figure.

2.2.3 Vision Transformers

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

Transformers were introduced by Vaswani et al. in 2017 [46], and revolutionized the deep learning field, surpassing Recurrent Neural Networks (RNNs) for Natural Language Processing (NLP) tasks [47, 46]. The design behind transformers is based on self-attention mechanism, allowing the model to weigh the significance of each part of input. As their design lack recurrence or convolutions, transformers use positional embeddings to represent the order of tokens in a sequence [46]. Later, the Vision Transformer (ViT), an adaptation of the transformer architecture for image processing, was introduced by Dosovitskiy et al. (2021) [31]. Here, images are represented as sequences including the class label as a learnable token for classification. The input image is divided into a sequence of patches, which are then flattened and liniarly 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 [31]. The illustration in figure 2.9 shows the architecture of Vision Transformers.

Unlike CNNs, ViTs have much less image-specific inductive bias [31]. While the inductive bias in CNNs help the model generalize to unseen data [40], the ViT architecture does not include built-in assumptions about the structure of the

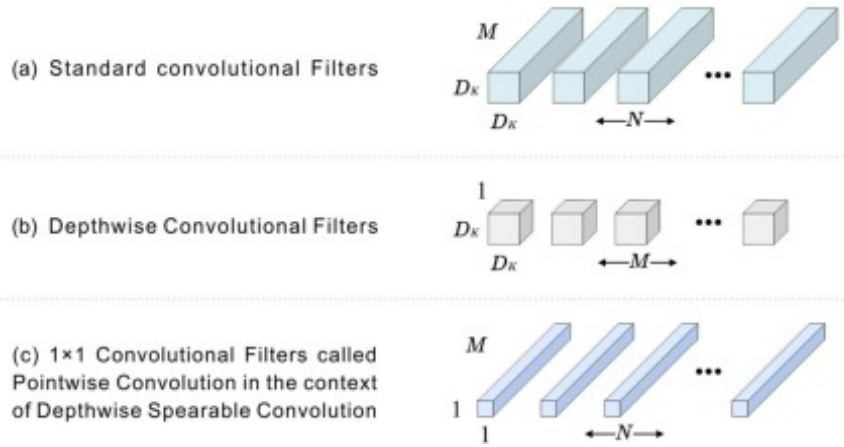

Figure 2.8: Depthwise Separable Convolution. Figure from `https://www.sciencedirect.com/topics/computer-science/depthwise-separable-convolution`. TODO: Make this figure.

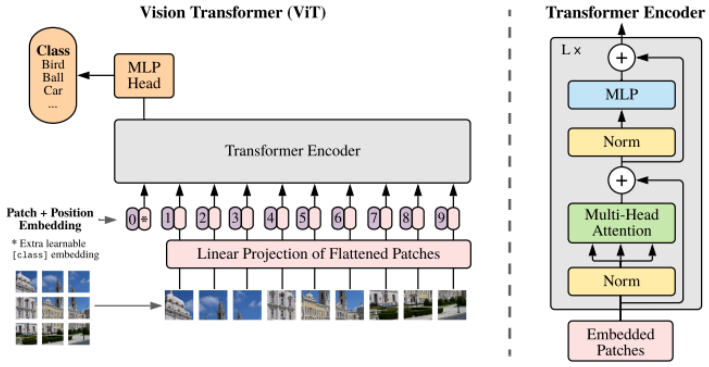

Figure 1: Model overview. We split an image into fixed-size patches, linearly embed each of them, add position embeddings, and feed the resulting sequence of vectors to a standard Transformer encoder. In order to perform classification, we use the standard approach of adding an extra learnable "classification token" to the sequence. The illustration of the Transformer encoder was inspired by Vaswani et al. (2017).

Figure 2.9: Vision Transformer architecture. Figure from [31]. TODO: Make this figure.

data, only the MLP layers are local an translationally equivalent [31]. As a result, ViTs require more training data to learn spacial relations compared with CNNs.

TODO: What made the ViT get SOTA? Why were they better than CNNs? Scalability. `https://www.youtube.com/watch?v=QqejVOLNDHA`.

ViT-B/16 Architecture

the ViT-B/16 is a Vision Transformer that leverages the transformer architecture for image classification [31], pre-trained on TODO: ImageNet-1K or ImageNet-21K?. 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 [31]. Additionally, each layer includes an MLP, and 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 [48].

ConvNeXt Base Architecture

ConvNext, introduced by Liu et al. in 2022 [49], 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. However, these are later combined in the final layers of the network.

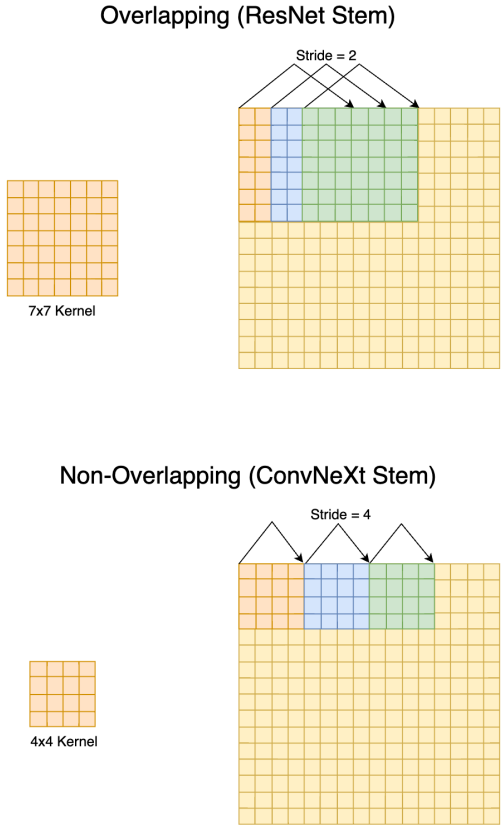

Figure 2.10: ResNet stem vs ConvNeXt stem. Figure from `https://www.kungfu.ai/blog-post/convnext-a-transformer-inspired-cnn-architecture` TODO: Make this figure.

Next, inspired by ResNeXt [50], the ConvNeXt use depthwise convolution, already described in the MobileNetV2 section 2.2.2, to increase the FLOPs/accuracy trade-off. Likewise, the ConvNext incorporates inverted bottleneck block, which are an important design in transformers as well as MobileNetV2 [49]. Additionally, the ConvNext architecure uses a $7time7$ depthwise convultion in each block on the condition of moving the depthwise convolution layer up. See figure 2.11.

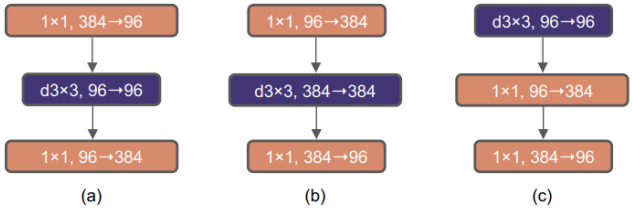

Figure 3. **Block modifications and resulted specifications. (a)** is a ResNeXt block; in **(b)** we create an inverted bottleneck block and in **(c)** the position of the spatial depthwise conv layer is moved up.

Figure 2.11: Block modifications. Figure from [49] TODO: Make this figure.

Instead of using the commonly used activation function in ResNets, the Rectified Linear Unit (ReLU), the ConvNeXt use Gaussian Error Linear Unit (GeLU). This activation function has shown to perform better than ReLU for transformers [49]. 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 with Layer Normalization. Finally, ConvNeXt introduces separate downsampling layers between blocks rather than implementing downsampling within the blocks themselves.

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. [5] class re-blancing 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 modification 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 [5]. 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 adresseses 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 [5], hence the need for more sophistacated approaches.

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 [5], as described later in this section. Consequently, tail classes are often misclassified. To address this imbalance, class-sensitive learning methods modify the loss function TODO: or how losses er calculated to pay more attention to minority classes, thus improving overall performance.

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 [5]. Classes with fewer samples are assigned greater weights, thus re-balancing the training process.

Re-Margining Adjusts the decision boundaries by introducing class-dependent margins, ensuring that minority classes have more room to establish discriminative features in the embedding space [5, 19].

Logit Adjustment Directly shifts the model's raw outputs using class priors [51, 52]. This approach normalizes the logits to account for imbalance in class distribution before computing the loss.

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

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 [32, 1].

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 [1]. 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. [53].

For image classification tasks, the standard loss function is the cross-entropy loss combined with the softmax activation function [32] 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:

$$P(y = i \mid \mathbf{z}) = \frac{\exp(z_i)}{\sum_{j=1}^{K} \exp(z_j)} \tag{2.3}$$

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 *Softmax Cross-Entropy (CE) loss* is a fundamental building block in training deep classifiers and is widely regarded as the baseline in classification tasks [5, 38, 54]. The CE loss combines the cross-entropy loss and the softmax activation function, and thus is commonly referred to as the *Softmax Loss* [32, 1, 5].

The CE loss measures the difference between the predicted probability distribution P and the true class label y, defined as [32, 38]:

$$\mathcal{L}_{\text{CE}} = -\sum_{i=1}^{K} y_i \log(P(y = i \mid \mathbf{z})) \tag{2.4}$$

Inserting the Softmax probabilites from equation (2.3) yields:

$$\mathcal{L}_{\text{CE}} = -\log\left(\frac{\exp(z_i)}{\sum_{j=1}^{K} \exp(z_j)}\right) \tag{2.5}$$

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 Softmax Loss (eq. (**??**)) 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 [51]. 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.

Re-Weighting

Weighted Softmax Cross-Entropy Loss The *Weighted Softmax Cross-Entropy (WCE) loss* modifies the standard CE loss (equation (**??**)) by introducing a weighting factor w. For multiclass classification, the CE loss it is multiplied by the inverse class frequencies $1/\pi_y$ [5, 55], given by:

$$\mathcal{L}_{\text{WCE}} = -w \log(p_y) = -\frac{1}{\pi_y} \log(p_y) \tag{2.6}$$

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

Focal Loss *Focal Loss* [55] 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. (**??**)). Here, $\gamma \geq 0$ is a tunable focusing parameter. The Focal Loss modifies the CE loss by applying the inverse prediction probability as follows:

$$L_{fl} = -(1 - p_y)^\gamma \log(p_y) \tag{2.7}$$

This factor increases the weight of misclassified examples, ensuring the model prioritizes learning from challenging samples. Figure 2.12 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 2.12 represents the standard cross-entropy loss for $\gamma = 0$.

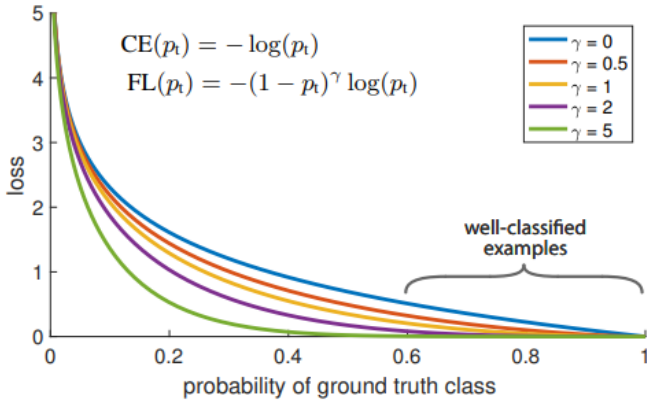

Figure 2.12: Figure from [55]. TODO: Make this figure

Class-Balanced Loss *Class-Balanced (CB) Loss* [27] 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 2.13, 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 $(1 - p)$.

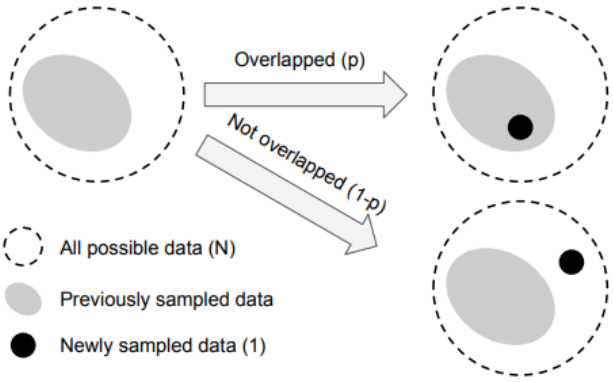

Figure 2. Giving the set of all possible data with volume N and the set of previously sampled data, a new sample with volume 1 has the probability of p being overlapped with previous data and the probability of $1 - p$ not being overlapped.

Figure 2.13: Figure from [27]. TODO: Make this figure

Cui et al. [27] 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:

$$E_n = \frac{1 - \beta^n}{1 - \beta} \tag{2.8}$$

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

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

$$L_{cb} = -\frac{1 - \beta}{1 - \beta^{n_y}} \log(p_y) \tag{2.9}$$

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 equation (2.8) grows large for minority classes (small n_y) and small for majority classes (large n_y). When $\beta = 0$ the loss is equivalent to the CE loss. Contrary, when $\beta \longrightarrow 1$ the re-weighting effect grows.

Equalization Loss *Equalization (EQ) Loss* [56] aims to mitigate the over-suppression of tail classes, which occurs when these underrepresented classes serve predominantly as negative examples for the more frequent classes. The idea is to

ignore the gradients for rare classes so that they are not excessively penalized when they appear as negatives, preventing their learned representations from being overshadowed. An analysis of gradient impact on classes is seen in figure 2.14.

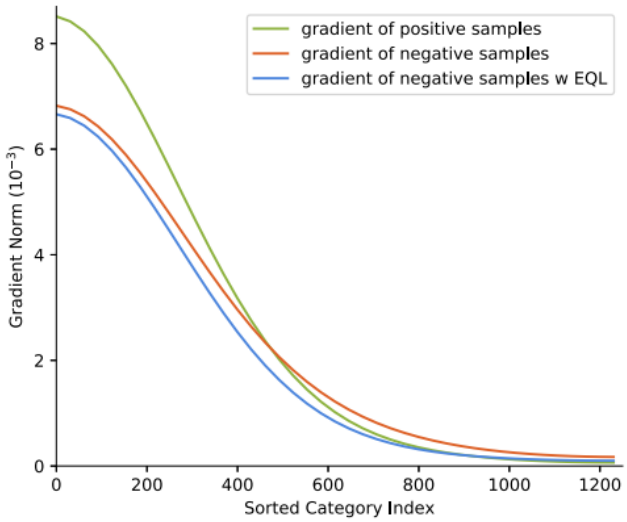

Figure 1: The overall gradient analysis on positive and negative samples. We collect the average L_2 norm of gradient of weights in the last classifier layer. Categories' indices are sorted by their instance counts. Note that for one category, proposals of all the other categories and the background are negative samples for it.

Figure 2.14: Gradient analysis. [56].

The EQ loss was designed for object recognition, but the authors decided to adopt the principles into image classification, officially called the *Softmax Equalization (SEQ) Loss*, defined as:

$$\mathcal{L}_{SEQL} = -\sum_{j=1}^{C} y_j \log(\tilde{p}_j) \tag{2.10}$$

where

$$\tilde{p}_j = \frac{e^{z_j}}{\sum_{k=1}^{C} \tilde{w}_k e^{z_k}} \tag{2.11}$$

and the weight term is given by:

$$\tilde{w}_k = (1 - \beta T_\lambda(f_k))(1 - y_k) \tag{2.12}$$

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.

Logit Adjustment

Logit adjustment is a class re-balancing technique that aims to optimize the class imbalance by adjusting the model outputs, i.e. prediction logits, typically based on prior class probabilities [52, 51]. TODO: Write about the label distribution shifts between training and test data.

The output model outputs are computed by the following equation:

$$\eta_j = \theta_j^T f(x) \tag{2.13}$$

where θ_j are the weights for the last layer for class j, and $f(x)$ is the feature extractor function [51].

Balanced Softmax Loss The *Balanced Softmax (BS)* Loss modifies the standard softmax loss (eq. (**??**)) by directly incorporating class priors π_y into the logits before computing probabilities [51]. 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.

In *Balanced Meta-Softmax for Long-Tailed Visual Recognition*, the authors represent Softmax regression as a multinomial distribution, ϕ, dependent on η, defined as:

$$\phi_j = \frac{e^{\eta_j}}{\sum_{i=1}^{k} e^{\eta_i}} \tag{2.14}$$

where k are the number of classes.

The Balanced Softmax uses the output logits η (eq. (2.13)) 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 [51].

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:

$$\hat{\phi}_j = \frac{n_j e^{\eta_j}}{\sum_{i=1}^{k} n_i e^{\eta_i}}. \tag{2.15}$$

where n_j is the number of samples in class j.

The authors proved that this method can accomodate the label distribution shifts between training and test sets [51]. ultimately, the Balanced Softmax loss function is becomes:

$$\mathcal{L}_{bs} = -\log\left(\frac{\pi_y \exp(z_y)}{\sum_j \pi_j \exp(z_j)}\right) \tag{2.16}$$

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.

Margin Modification (Re-Margining)

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

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. Figure 2.15 illustrates how the margin affects the decision boundary for binary classification. As the decision boundary separates the regions of input space corresponding to different class predictions, by increasing the margin, the model becomes more confident in classification.

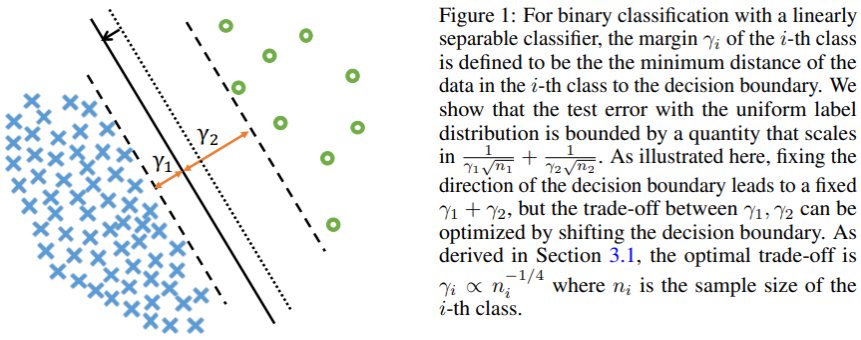

Figure 2.15: Margins [19]. TODO: make this figure.

The margin is defined as:

$$\gamma(x, y) = f(x)_y - \max_{j \neq y} f(x)_j \tag{2.17}$$

which is the distance between the prediction of the sample (x, y) and the decision boundary.

The training margin for class j is defined as:

$$\gamma_j = f(x)_y - \max_{i \in S_j} \gamma(x_i, y_i) \tag{2.18}$$

The classical generalization bound is defined as

$$\text{imbalanced test error} \lesssim \frac{1}{\gamma_{min}} \sqrt{\frac{C(\mathcal{F})}{n}} \tag{2.19}$$

In a balanced setting:

$$\frac{1}{2}\text{err}[\text{class1}] + \frac{1}{2}\text{err}[\text{class2}] \lesssim \left(\frac{1}{\gamma_1\sqrt{n_1}} + \frac{1}{\gamma_2\sqrt{n_2}}\right)\sqrt{C(\mathcal{F})} \tag{2.20}$$

This boundary can be shifted by taking the minimum: $\min \gamma_1\sqrt{n_1} + \frac{1}{\gamma_2\sqrt{n_2}}$, such that $\gamma_1 + \gamma_2 = C$. This yields the optimal solution:

$$\gamma_i \propto n_i^{-1/4} \tag{2.21}$$

with the standard solution being $\gamma_1 = \gamma_2$ [19].

LDAM Loss The *Label-Distribution-Aware Margin (LDAM)* Loss [19] 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 equation (2.21) by increasing the distance between the largest and secon largest logit for each class and introducing that to the cross-entropy loss (eq. (**??**)).

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

$$\gamma_j = \frac{C}{n_j^{1/4}} \tag{2.22}$$

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 TODO: reference, will encourage the model to respect the margins in equation (2.22):

$$\mathcal{L}_{ldam-hg} = \max\left(\max_{j\neq y}\{z_j\} - z_y + \Delta_y, 0\right) \tag{2.23}$$

where

$$\Delta_j = \frac{C}{n_j^{1/4}} \quad \text{for } j \in \{1, \ldots, k\}. \tag{2.24}$$

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 equation (2.23) 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:

$$L_{ldam} = -\log\left(\frac{\exp(z_y - \Delta_y)}{\sum_j \exp(z_j - \Delta_j)}\right) \tag{2.25}$$

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 [24] 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 [5]. This adjustment ensures that the experiments simulate real-world class imbalance.

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

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

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

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

3.2 Dataset Preparation and Specifications

A description of this section here.

Following the dataset structure used in *Deep Long-Tailed Learning: A Survey*, the CIFAR100 dataset was modified to create a long-tailed training set and a balanced test set.

3.2.1 Benchmark Dataset Selection

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

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

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

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

TODO: Describe the type of images in CIFAR-100 and include examples.

3.2.2 Data Characteristics: Class Distribution

Understanding the class distribution in long-tailed datasets is essential for evaluating the impact of the class imbalance on model performance. This section

analyzes the class distribution of the benchmark ImageNet-LT used in [5], which serves as a reference for creating a similar distribution across the CIFAR-100-LT training, evaluation, and test datasets.

The ImageNet-LT dataset, introduced by Liu et al. [4], has been widely used in empirical studies, including *Deep Long-Tailed Learning: A Survey* [5]. The class distributions of its training, validation, and test sets are shown in Figures 3.1, **??**, and **??**, respectively. The ImageNet-LT is constructed from the original ImageNet-2012 following the Pareto-distribution (Eq. (2.2)) with power value $\alpha = 6$ [4], resulting in an imbalance ratio of 256, meaning that the most frequent class has 256 times more samples than the least frequent class. As suggested in figure 3.1, the most frequent class has 1280 images, and the least frequent class 5 images [4]. The power value of 6 is relatively large, resulting in the steep imbalance and very few samples in the tail, as seen in figure 3.1. The underrepresentation in the tail classes poses a significant challenge for models to learn from tail classes. Both the validation and test sets are balanced, as shown in figures **??** and **??**, respectively. There are 20 samples per class in the validation set, and 50 samples per class in the test set. TODO: remove the figures, and just leave it in writing.

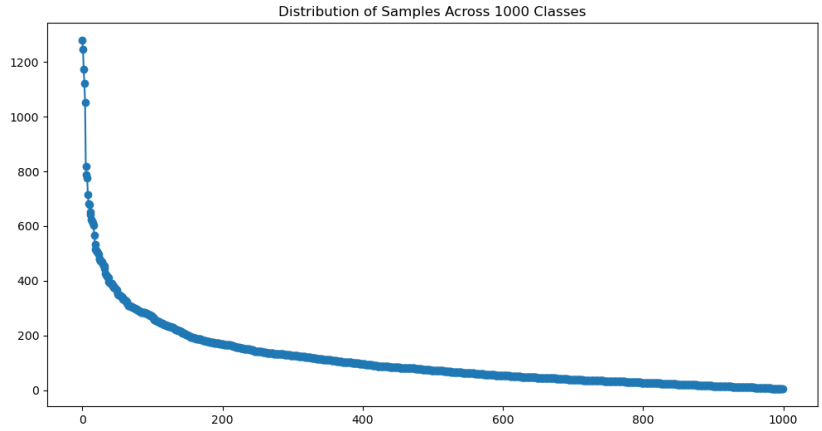

Figure 3.1: The class distribution of the training images for the ImageNet-LT dataset shows a long-tailed distribution.

However, investigating the CIFAR-100-LT dataset used in *Learning Imbalanced Datasets with Label-Distribution-Aware Margin Loss* by Cao et al. [19] shows that the training dataset follows a exponential decay, and not a pareto distribution. The imbalance factor in their studies is 100, meaning that the most frequent class has 100 times more samples than the least frequent class, following equation (2.1). This means that the imbalance is not as steep as with the Pareto distribution, resulting in a middle section of classes with fewer samples than the head classes, but more samples than the tail classes, see figure 3.2. 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 the PyTorch torchvision.datasets.CIFAR100 [57]. The training and test datasets were preprocessed by converting the images to tensors using the ToTensor transformation and saved as .pth files for efficient loading during experiments.

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

Table 3.2: Class Distributions Used in Experiments

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

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 [19], the imbalance was created using exponential decay. Here, the number of samples per class decreases exponentially, controlled by the imbalance factor, following equation (2.1). For this thesis, an imbalance factor of 100 was applied. This means that the most frequent class contains a 100 times more samples than the least frequent class.

The resulting class distribution varied from the most frequent class having 450 samples to the least frequent class having only 4 samples, as shown in figure 3.2. 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

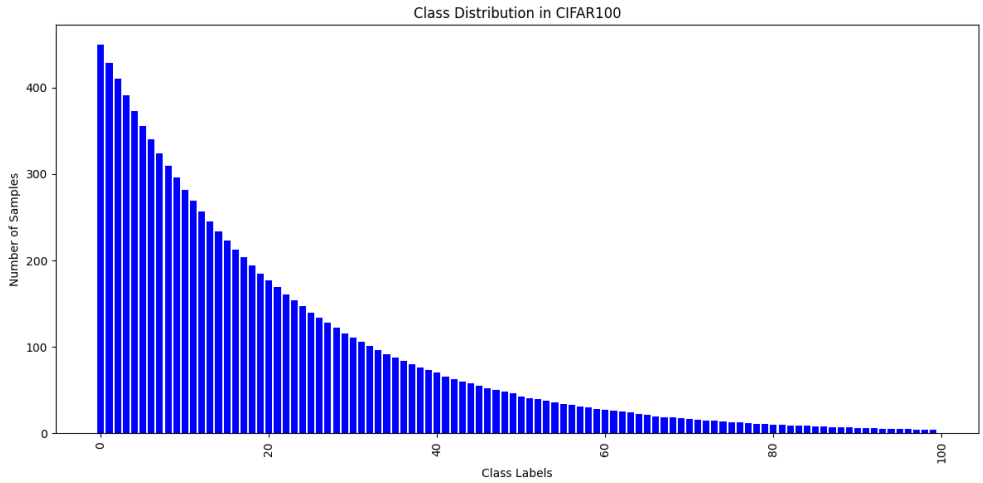

Figure 3.2: The class distribution of the CIFAR-100 long-tailed training set with 450 samples in the most frequent class.

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 3.3. The dataset was saved locally to maintain reproducibility.

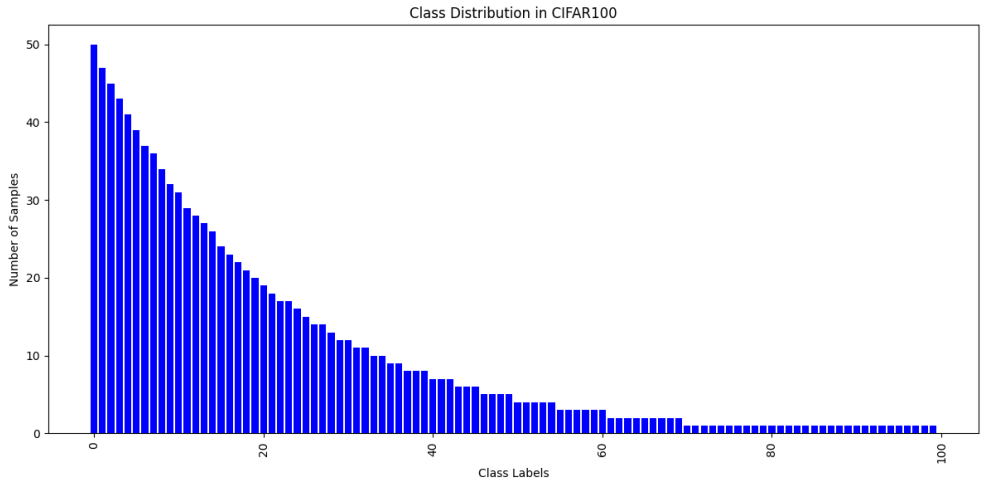

Figure 3.3: The class distribution of the CIFAR-100 long-tailed test set with 50 samples in the most frequent class.

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

3.2.4 Data Augmentation

Data augmentation and normalization were applied to the CIFAR-100 dataset to improve the generalization capability of the models. For the CIFAR-100 dataset, the augmentations were applied using the CIFAR-100 RGB statistics, with mean values [0.4914, 0.4822, 0.4465], and standard deviations [0.2023, 0.1994, 0.2010] [25]. These values are used to normalize the images after applying the transformations. However, the models used in this study are all pretrained on ImageNet, and choosing the ImageNet RGB statistics could potentially influence the training, as the pretrained weights are optimized for inputs with ImageNet statistics. Future work could explore the impact of using ImageNet statistics instead.

Images were resized to 224x224 pixels to match the input requirements of the ImageNet-pretrained models. Random cropping with a padding of 4 was applied to introduce spatial variations, while horizontal flipping with a 50% probability further augmented the data. After these transformations, images were normalized using the CIFAR-100-specific mean and standard deviation values, ensuring consistency with the dataset.

For validation, a simpler approach was used, where images were resized to 224x224 pixels and normalized using the same mean and standard deviation values as the training set.

3.3 Long-tailed Learning Techniques

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 [58]. 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 [59, 60, 61]. 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: Models. TODO: caption

Model	Type	Pre-trained	Parameters (millions)	top-1 accuracy
MobileNetV2 [28]	CNN	ImageNet-1K	-	72.154 [62]
ResNet50 [29]	CNN	ImageNet-1K	-	80.858 [44]
ConvNeXt-Base [30]	CNN	ImageNet-1K	88.591.464	84.062 [63]
ViT-B/16 [31]	ViT	ImageNet-21K (ft on ImageNet-1K)	-	-

ResNet-50 Resnet-50 [29] is a convolutional neural network designed with residual connections to mitigate the vanishing gradient problem, as detailed in section 2.2.2. Introduced by He et al. in 2015, the ResNet architecure achieved unprecedented results and won ILSVRC 2015 for image classification tasks [58].

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 [29]. ResNet-50 has been utilized in various domains of image classification, including tasks such as medical imaging [64, 65] and face detection [66], and commonly serves as a baseline in research [13, 67, 12, 52]. Leveraging pretrained weights on ImageNet-1K, it achieves better generalization on unseen datasets [68, 69, 70, 71], 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 [29]. This balance of performance and efficiency makes it suitable for smaller datasets like CIFAR-100 [72].

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 [29, 13].

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 2.2.2. This architecture was chosen to represent an example of a modern and lightweight CNN architecture that has achieved state-of-the-art performance [28], and has shown great performance in image classification tasks, including medical imaging [73] and fruit classification [74, 75]. 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 [28].

ViT-B/16 The Vision Transformer (ViT-B/16) [31] 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 16x16 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 [31].

The implementation of the ViT-B/16 was done through the timm library [76], trained on ImageNet-21K and fine-tuned on ImageNet-1K. However, for consistency and direct comparison, the implementation of ViT-B/16 through the PyTorch library [48], pre-trained on ImageNet-1K, would be preferred. Especially since the none of the layers are frozen during training, allowing for fine-tuning on the CIFAR-100 dataset. This has potentially influenced the performance of the model. However, since this is a vision transformer, the model benefits from pretraining on a larger dataset and then fine-tuned due to the lack of inductive bias [31, 77], as discussed in section 2.2.3.

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 [31], and has been used in image classificataion tasks, such as brain tumor classification [78]. 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 [49] 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 2.2.3. These innovations allow ConvNeXt to achieve state-of-the-art performance in image classification tasks while retaining the core efficiency of convolutional operations [49]. In its original paper [49], 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 [49]. 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 [5], 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 [5], which serves as a foundational baseline in this study. However, this loss inherently favors head classesl 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 [5, 55], 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 detail 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) [5] 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 sec-

tion 2.4.2. 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 [5]. 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 [27], 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 [5, 27]. 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 2.4.2. 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 [5]. 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 [27].

Balanced Softmax Loss The Balanced Softmax Loss [51] 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 2.4.2.

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 [51].

Focal Loss Instead of using training labels, Focal Loss [55] modifies the CE loss by including a focusing parameter that down-weights well-classified exam-

ples, 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 2.4.2. 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 [55, 5]. 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 [5]. Over time, this can discourage the network from correctly identifying the minority classes. Equalization Loss [56] 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 2.4.2. This strategy highlights a subtlety in the re-weighting paradigm: it is not only about improve 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 [5]. 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 [5]. 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 2.4.2. Label-Distribution-Aware Margin (LDAM) [19] 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 [19]. 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 [19].

The paper *Learning Imbalanced Datasets with Label-Distribution-Aware Margin Loss* 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 [19]. 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.

Summary

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 [54]	$\mathcal{L}_{ce} = -\log(p_y)$	-
Focal loss [55]	$\mathcal{L}_{fl} = -(1-p_y)^{\gamma}\log(p_y)$	re-weighting
Weighted Softmax loss [5]	$\mathcal{L}_{wce} = -\frac{1}{\pi_y}\log(p_y)$	re-weighting
Class-balanced loss [27]	$\mathcal{L}_{cb} = -\frac{1-\gamma}{1-\gamma^{n_y}}\log(p_y)$	re-weighting
Balanced Softmax loss [51]	$\mathcal{L}_{bs} = -\log\left(\frac{\pi_y \exp(z_y)}{\sum_j \pi_j \exp(z_j)}\right)$	logit adjustment
Equalization loss [56]	$\mathcal{L}_{eq} = -\log\left(\frac{\exp(z_y)}{\sum_j \omega_j \exp(z_j)}\right)$	re-weighting
LDAM loss [19]	$\mathcal{L}_{ldam} = -\log\left(\frac{\exp(z_y-\Delta_y)}{\sum_j \exp(z_j-\Delta_j)}\right)$	re-margining

TODO: add results from the original papers in a table

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 [44, 62, 79] except for ViT-B/16 that used the ImageNet-21K pretrained weights and ImageNet-1K fince-tuning from the timm library [76]. The fully connected layer was replaced with 100 classes to match the CIFAR-100 dataset. The specifications can be seen in table B.4. Each model is trained with a batch size of 128 and the Adam optimizer [53] 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 [80]. 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 B TODO: fix this reference for further specifications.

Optimizer

The Adam optimization algorithm [53] is a popular optimizer in deep learning tasks, including image classification [81, 82]. Despite the Stochastic Gradient Desect (SGD) 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 [53]. 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 in better perfomance on small datasets like CIFAR-100 [52]. Future work could include training with the SGD with fine-tuning of hyperparameters.

This study implements the PyTorch Adam optimizer [83] 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 [38, 84, 81]. 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 [84].

However, this approach comes with trade-offs. Fine-tuning all layers increases the risk of overfitting [38], 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 [38], as the lower layers learn more generic features, like edges and shapes, and the top layers learn more specific features common for the dataset [85]. 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 [5]. It measures the proportion of samples where the model's top prediction matches the ground truth label, and is given by [86]:

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

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 TODO: reference?, especially in tasks involving class imabalance. 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) TODO: references, 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.

TODO: Compare the different types of losses against each other. See [52].

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 ResNet50 architecture (Acc1: 0.6053), closely followed by the ConvNeXt Base architecture (Acc1: 0.5789). However, this improved tail-class performance comes at the cost of head-class accuracy, where ConvNeXt Base outperforms ResNet50 with a top-1 accuracy of 0.8685 compared to 0.8270. Overall, ConvNeXt Base demonstrates better performance across all classes (Acc1: 0.8230) compared to ResNet50 (Acc1: 0.7916). See Tables 4.4 and 4.8 for reference. These findings, however, require further statistical analysis to confirm their significance and that the observed differences are not simply due to random variability.

Class-Balanced Loss consistently underperformed, warranting further investigation into its implementation. Similarly, the ViT-B/16 architecture demonstrated the lowest overall accuracy when trained on both balanced and long-tailed datasets (Acc1: 59.06 %, see Table 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 [87]. This discrepancy suggests that the ViT-B/16 configuration used for the experiments in this thesis may not be well-suited for smaller-scale datasets such as CIFAR-100 without careful optimization as discussed in sections 2.2.3 and 3.3.1 TODO: write these sections.

4.2 Overall Results

This section presents the overall results of all experiments conducted in this thesis, commenting on the best and worst performance of each loss design on a given model, but not directly comparing loss designs or models. It provides an overview of all findings to understand how each combination of architecture and loss function behaves.

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

4.2.1 MobileNetV2

Results from Balanced Training

Table 4.1: Top-1 accuracy results for MobileNetV2 on the balanced dataset across all loss functions.

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

From Table 4.1, Balanced Softmax Loss achieves the best accuracy on the balanced test dataset (Acc1: 0.8034) and the highest accuracy on tail classes (Acc1: 0.9211). Not surprisingly, Softmax, Weighted Softmax, and Class-Balanced Loss yield the same accuracies across all test datasets due to shared cross-entropy-based designs that become indistinguishable when trained on balanced data according to equations (**??**), (2.6), and (2.9) described in section 2.3. Overall, all methods achieve comparable results to the softmax baseline, while outperforming the best published result for MobileNetV2 trained with CIFAR-100, as seen in table 4.9.

TODO: This paragraph could potentially be moved to an overall discussion section, as it includes a discussion of other model behavior. Although LDAM Loss demonstrates lower overall accuracy, its high tail-class accuracy matches that of Balanced Softmax and Equalization Loss, suggesting that tail-class performance may have reached a saturation point due to the limited number of samples in tail classes. This notion is further supported by identical performance across multiple long-tailed subsets with other backbones (see tables 4.3 and 4.7), suggesting potential limitations in class-specific data quality or quantity. These limitations could be investigated through testing on a larger dataset.

Results from Long-Tailed Training

Table 4.2: Top-1 accuracy results for MobileNetV2 on the long-tailed dataset across all loss functions.

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

From Table 4.2, Balanced Softmax Loss outperforms other losses in terms of balanced test accuracy (0.5796) and especially tail classes (0.4211). Although Softmax and Focal Loss surpass Balanced Softmax on the long-tailed dataset and head classes, the improvements Balanced Softmax provides for tail and middle classes are notable. Class-Balanced Loss shows an overwhelming underperformance compared with the balanced training in table 4.1, and now appears severely limited, possibly due to implementation issues that must be investigated TODO: investigate. The difference in performance patterns compared to the softmax baseline shows that loss functions designed for imbalance can provide more nuanced performance gains.

4.2.2 ResNet50

Results from Balanced Training

Table 4.3: Evaluation results for ResNet50 trained on the custom balanced dataset, showing Acc1.

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

From Table 4.3, Softmax, Weighted Softmax, and Class-Balanced Loss yield identical performance when trained on balanced data due to their cross-entropy term

in equations (2.9), (2.6), and (2.9). Balanced Softmax Loss demonstrates strong performance on the long-tailed test set (0.8430) and head classes (0.8460), while still performing comparably on tail classes. Overall, the performance of the class-rebalancing methods compare to that of the softmax baseline. The best performing design yield an accuracy of 83.24%, not far behind the best published result for ResNet50 with an accuracy of 86.90% as seen in table 4.9.

Results from Long-Tailed Training

Table 4.4: Evaluation results for ResNet50 trained on the long-tailed dataset, showing Acc1.

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

From Table 4.4, Balanced Softmax Loss significantly improves tail-class accuracy (0.6053), outperforming other losses, while also outperforming on balanced and middle sets. Softmax Loss achieves the best performance on the long-tailed dataset and head classes, indicating that while Balanced Softmax improves minority-class accuracy, Softmax remains strong for majority classes. Class-Balanced Loss and LDAM Loss both show underwhelming results, pointing toward the necessity of further investigation for Class-Balanced Loss TODO: investigate or combining LDAM with DRW [19].

4.2.3 ViT-B/16

Results from Balanced Training

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, see table 4.9. This suggests that further adjustments are needed, possibly including extended training time, data augmentation, or different optimizers, as ViT architecture differs from CNNs, as describe in section 2.2.3.

Results from Long-Tailed Training

From Table 4.6, performance remains low across all loss functions. Balanced Softmax Loss slightly improves results on balanced and middle categories, while

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	0.5620	0.5671	0.5521	0.6036	0.7368
Balanced Softmax loss	0.5628	0.5642	0.5640	0.5325	0.7105
Equalization loss	0.5634	0.5519	0.5462	0.5503	0.6842
LDAM loss	**0.5906**	**0.6013**	**0.5924**	**0.6095**	**0.7632**

Table 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	0.0558	0.0076	0.0000	0.0237	**0.1053**
Balanced Softmax loss	**0.2460**	0.4244	0.4822	**0.2130**	0.0789
Equalization loss	0.2168	0.4215	0.4893	0.1716	0.0263
LDAM loss	0.1570	0.2750	0.3140	0.1361	0.0263

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

4.2.4 ConvNeXt Base

Results from balanced Training Dataset

From Table 4.7, Balanced Softmax Loss achieves the best balanced test accuracy (0.8364) while the Softmax architechtures tie for top performance on the long-tailed dataset, head classes, and tail classes. The saturation of tail-class performance is likely due to the quality or quantity of tail classes. Balanced Softmax's advantage on balanced data may stem from its logit-adjustment mechanism, which can enhance generalization even when classes are balanced [51] TODO: investigate. Although the best published result for a ConvNext architecture trained with CIFAR-100 is the Conv2NeXt with an accuarcy of 83.82% (see table 4.9), the best perfomance of on the balanced test set is in alignment, yielding an accuracy of 83.64%.

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	0.8332	**0.8535**	**0.8566**	0.8166	**0.9474**
Balanced Softmax loss	**0.8364**	0.8344	0.8365	0.7988	**0.9474**
Equalization loss	0.8318	0.8468	0.8448	**0.8343**	**0.9474**
LDAM loss	0.8316	0.8373	0.8412	0.8047	0.8947

Results from Long-Tailed Training Dataset

Table 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	0.0142	0.0019	0.0000	0.0000	0.0526
Balanced Softmax loss	**0.6460**	0.8230	0.8685	0.6509	**0.5789**
Equalization loss	0.5956	0.8278	0.8768	**0.6923**	0.3421
LDAM loss	0.3770	0.5956	0.6445	0.4260	0.2632

From Table 4.8, Balanced Softmax Loss outperforms on the balanced test set (0.6460) and tail classes (0.5789) by a considerable margin. Softmax achieves the best performance on the long-tailed dataset and head classes, while Equalization Loss leads on middle classes. The poor performance of Class-Balanced Loss stands out, calling for further investigation into its implementation TODO: investigate.

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 or similar 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, ResNet50 exhibits a slightly lower accuracy compared to benchmarks, which may be explained by differences in optimization strategies, such as the use of Adam optimizer instead of SGD [52].

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

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

Table 4.9: Comparison of model performance on balanced CIFAR-100 with published benchmarks.

Model	**Result (Acc1)**	**Benchmark Result (Acc1)**
MobileNetV2	80.34%	73.20% [88]
ResNet50	83.24%	86.90% [89]
ViT-B/16	59.06%	93.51% [84]
ConvNeXt Base	83.64%	94.04% [84]

4.4 Performance Comparisons

The models are compared by taking the mean of the results of the loss functions for each model. Excluding the generally underperforming Class-Balanced Loss, the mean and standard deviation across all loss designs for each model is plotted in figure 4.1. TODO: Include a table of the values. Here, the performance of the models MobileNetV2, ResNet50, ViT-B/16, and ConvNeXt Base is shown across the evaluation categories: balanced, long-tailed, head, middle, and tail. Each model is slightly offset along the x-axis to avoid overlap. Each point represents the mean accuracy of a model for a specific category and the bars represent the standard deviation. Again, the mean and standard deviation excludes those of the Class-Balanced Loss. The error bars indicate the variability in accuracy across different loss functions for each model and category. A longer error bar demonstrate a less consistent performance, dependent on the chosen loss function, while shorter error bars indicate consistant performance adhering to the specific model.

From figure 4.1, the shorter errorbars for MobileNetV2 indicate that its architecture may be more robust to the type of re-weighting strategy used than other architechtures. Contrary, the ResNet50V2 performance on tail classes shows greater sensitivity to the chosen loss design. ViT-B/16 shows a consistently lower mean accuracy, indicating that transformer-based approaches may require additional adaptations for smaller-scale or imbalanced datasets, as discussed in 3.3.1. ConvNeXt Base maintains strong and stable performance across several categories, benefiting from Balanced Softmax Loss as seen in table 4.8.

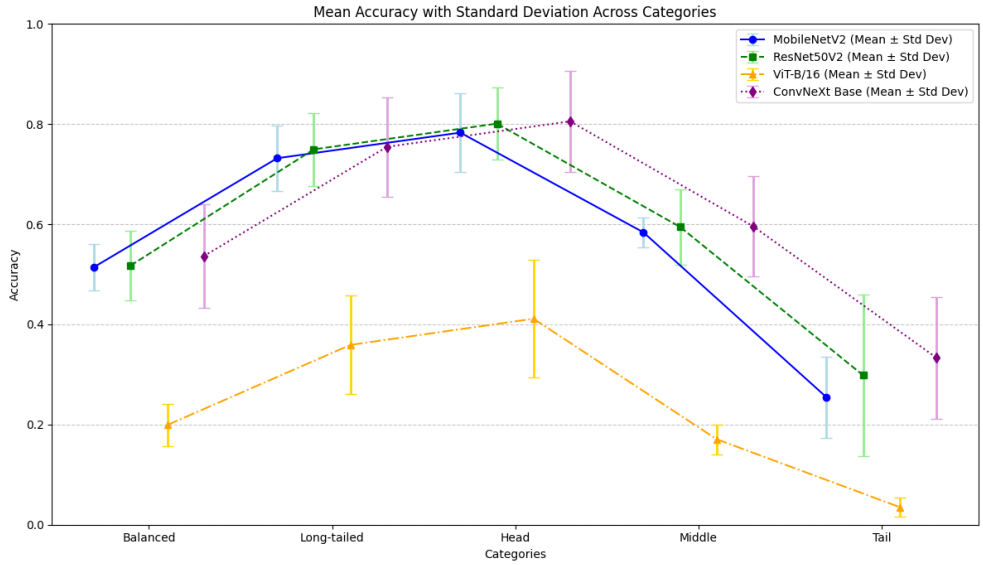

Figure 4.1: Mean Accuracy with Standard Deviation Across Categories for MobileNetV2, ResNet50, ViT-B/16, and ConvNeXt Base trained with CIFAR-100-LT. These values are excluding those of the Class-Balanced Loss.

The Balanced Softmax Loss emerges as the overall best performing loss design across all models with the least variance in performance across test categories, as figures 4.2, 4.3, 4.5, and 4.4 illustrate.

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

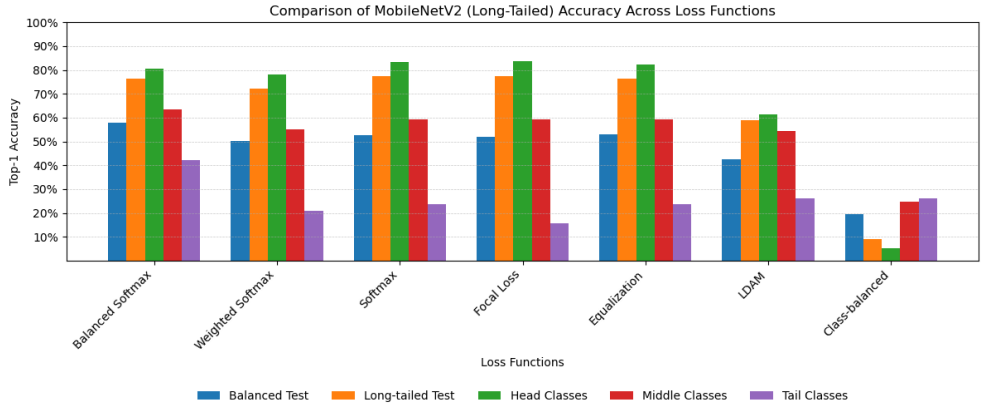

Figure 4.2: MobileNetV2 top 1 accuracy across test categories (Balanced, Long-tailed, Head, Middle, Tail) for different loss functions.

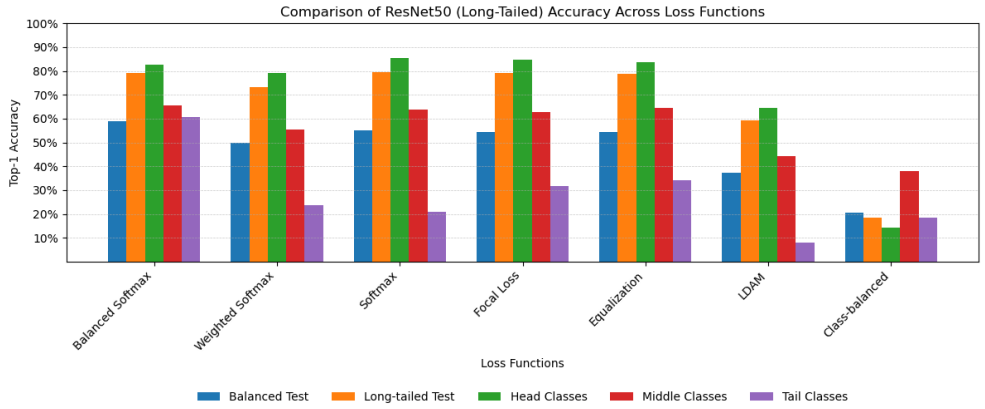

Figure 4.3: ResNet50 top 1 accuracy across test categories (Balanced, Long-tailed, Head, Middle, Tail) for different loss functions.

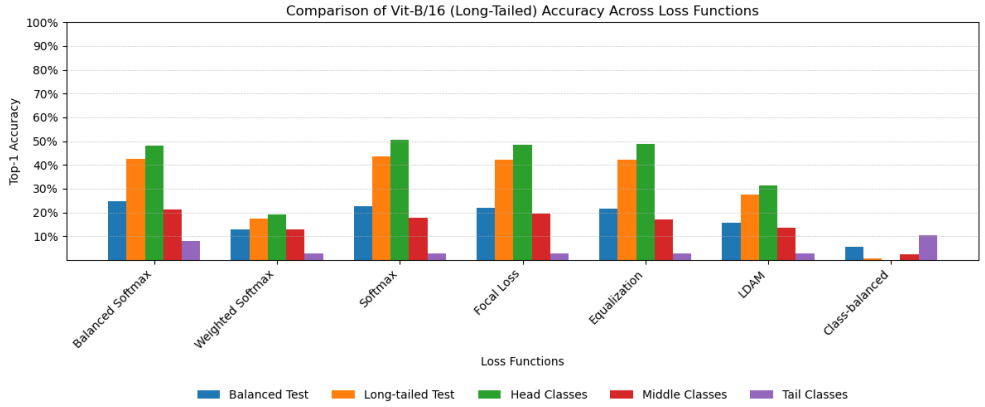

Figure 4.4: ViT-B/16 top 1 accuracy across test categories (Balanced, Long-tailed, Head, Middle, Tail) for different loss functions.

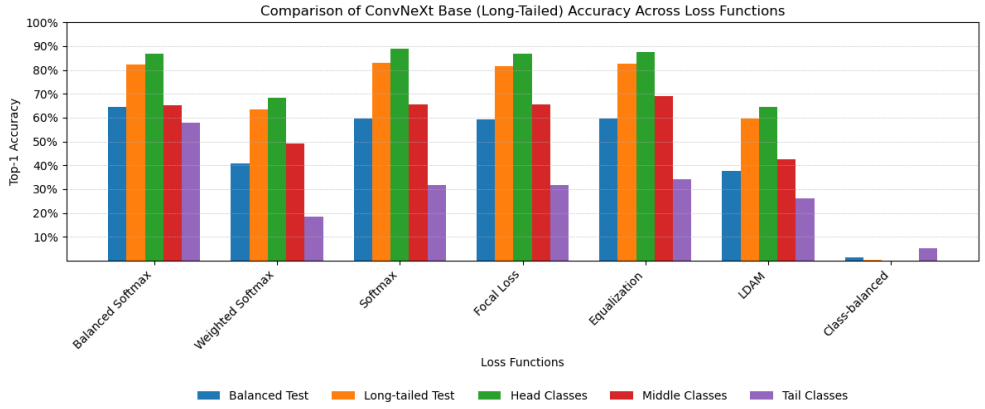

Figure 4.5: ConvNeXt Base top 1 accuracy across test categories (Balanced, Long-tailed, Head, Middle, Tail) for different loss functions.

This result is further underlined from figure 4.6, where the mean and standard deviation of the loss functions across ResNet50, MobileNetV2, ViT-B/16, and ConvNeXt is calculated and plotted. Here, the Balanced Softmax Loss shows a better average perfomance on tail classes, while also displaying the smoothest variation across evaluation categories TODO: calculate variance, ignoring the Class-Balanced loss. However, the standard deviation suggests that the perfomance of the Balanced Softmax Loss on tail classes depend on the model architecture, as earlier discussed. Table 4.10 display the exact values of the mean and standard deviations shown in figure 4.6.

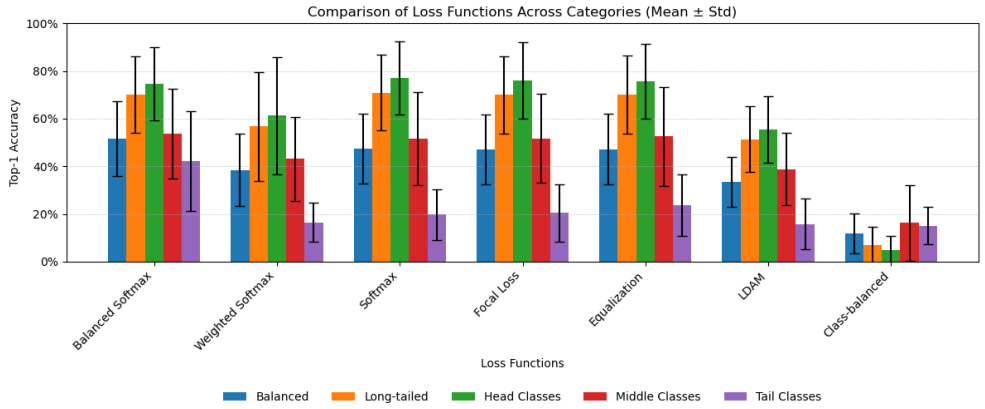

Figure 4.6: Performance trends of different loss functions across evaluation categories. Error bars indicate the standard deviation of accuracy across models, highlighting variability in performance for each loss function.

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

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

4.5 Summary and Discussion

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 competing results on other test categories. Furthermore, these findings highlight the impact of model architecture combined with loss designs, as the standard deviation demonstrates when comparing figures 4.1 and 4.6. Here, the longer errorbars suggests the dependency of loss design and model architecture, respectively. Further investigation of model and loss design combinations could lead to greater tail-class performance.

Although the CB Loss' overall superior performance on tail classes, this performance accuracy often came with trade-offs in performance on head classes. For example, while ResNet50 achieved the highest top 1 accuracy of 60.53% on tail classes using Balanced Softmax, compared that of ConvNeXt Base (57.89%), its head-class accuracy (82.70%) lagged behind ConvNeXt Base (86.85%). This trade-off highlights the challenge of optimizing for both head and tail classes simultaneously and indicates that Balanced Softmax prioritizes tail-class adjustments at the expense of head-class performance. These results, however, should be challenged by multiple runs to check for statistical significance.

Class-Balanced Loss, in contrast, consistently underperformed across all models and datasets. This discrepancy between its intended purpose and observed results suggests possible issues with its implementation TODO: investigate. Further investigation is necessary to understand these limitations and determine whether its performance can be improved. TODO: Make a table with original findings for comparison.

Likewise, the ViT-B/16 architecture underperformed significantly across all loss functions, categories, and training datasets, with the best accuracy of 59.06% on a balanced training dataset compared to its benchmark of 93.95% TODO: reference. These results indicate that the default ViT-B/16 architecture may not be well-suited for long-tailed datasets without further optimization [52, 82]. These could include a carefully chosen optimizer, longer training, or larger dataset. Comparatively, the CNN architectures displayed overall better performance with the same setup. TODO: here's another reference `https://arxiv.org/pdf/2205.01580v1`.

TODO: Include discussion of the choice of adam optimizer over the SGD.

TODO: Include a discussion of the implementation of the EQ Loss.

The performance of the LDAM could potentially improve by including the deferred re-weighting scheme (DRW), also introduced by Cao et al. [19]. According to related work [52], the DRW significanlty improves performance on long-tailed datasets in contrast to relying solely on LDAM for effective class re-balancing. TODO: reference benchmark performance.

The decision to derive test data from the training set, rather than splitting the validation set to match the class distribution to that of the ImageNet-LT (see section 3.2), could also influence the results, since the training data would include more samples. Likewise, the performance of the combinations of model architectures and loss designs could benefit from training with a larger dataset, like ImageNet and ImageNet-LT. Furthermore, these larger long-tailed benchmarks follow the Pareto distribution rather than an exponential decay, mirroring a natural occuring long-tailed dataset, as discussed in sections 2.1 and 3.2.

While Balanced Softmax stood out as the most robust loss function overall with the smallest variance across categories TODO: make this, statistical validation is required to confirm the significance of these findings by running multiple training sessions.

Overall, these findings emphasize the importance of aligning loss functions with both dataset characteristics and model architectures to address the challenges of

deep long-tailed learning. They also highlight the need for a nuanced approach that balances improvements in tail-class accuracy with minimal trade-offs in head-class and overall performance.

TODO: Adam optimization: Figure 1: Mean and standard deviation over 5 runs of per-class weight norms for a ResNet-32 under momentum and Adam optimisers. We use long-tailed ("LT") versions of CIFAR-10 and CIFAR-100, and sort classes in descending order of frequency; the first class is 100 times more likely to appear than the last class. Both optimisers yield solutions with comparable balanced error. However, the weight norms have incompatible trends: under momentum, the norms are strongly correlated with class frequency, while with Adam, the norms are anti-correlated or independent of the class frequency. Consequently, weight normalisation under Adam is ineffective for combatting class imbalance [52]. adam optimizer: `https://medium.com/@weidagang/demystifying-the-adam-optimizer-in-machine-learning-4401d162cb9e` Useful for large datasets [53]

Chapter 5

Conclusion and Future Work

The findings from the experiments conducted in this thesis underlines the difficulty of achieving strong performances with deep long-tailed learning. Specifically, the results emphasizes the importance of thoughtfully combining model architectures, loss design, and choice of configurations. Among the evaluated approaches, the Balanced Softmax Loss consistently achieved the best performance on tail classes, while maintaining a comparable performance on other categories across multiple CNN backbones. In contrast, Class-Balanced Loss underperformed across all models and categories, suggesting a fault in implementation TODO: investigate. Likewise, ViT-B/16's performance gap from both CNN-based architectures and its own benchmark imply that vision transformer architectures may require extensive adaptations to reach their full potential.

5.1 Revisiting the Goals of the Thesis

TODO: Add the goal from the introduction section and answer them below for the sake of the reader. 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 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, although the 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.

Secondly, the evaluations of different model architectures, namely the CNN-based and Vision Transformers, in combination with different class re-balancing designs has revealed that model design has an effect on the performance of long-tailed learning, and that the MobileNetV2 architecture exhibits greater robustnes against choice of loss design than other architectues evaluated in this thesis.

Finally, 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 Future Work

Future investigation would benefit from evaluating experiments with larger datasets, such as ImageNet or iNaturalist to gain more insight into the performance of models and long-tailed techniques. 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. Finally, other techniques, such as re-sampling in combination with class re-balancing, should be explored.

Bibliography

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

[2] 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.

[3] 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.

[4] 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.

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

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

[7] 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.

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

[9] 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.

[10] 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.

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

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

[13] 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.

[14] 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.

[15] 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.

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

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

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

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

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

[27] 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.

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

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

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

[31] 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.

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

[33] 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.

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

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

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

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

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

[39] Aman Arora. *Understanding Logits, Sigmoid, Softmax, and Cross-Entropy Loss in Deep Learning*. Accessed: 2024-12-31. 2024. URL: https://wandb.ai/amanarora/Written-Reports/reports/Understanding-Logits-Sigmoid-Softmax-and-Cross-Entropy-Loss-in-Deep-Learning--VmlldzoONDMzNTU3.

[40] Christopher Kim. *What is the Inductive Bias in Machine Learning?* https://medium.com/@chkim345/what-is-the-inductive-bias-in-machine-learning-212a5f53e9aa. Accessed: 2024-12-18. 2020.

[41] 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.

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

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

[44] TorchVision Contributors. *ResNet Implementation in TorchVision*. https://github.com/pytorch/vision/blob/main/torchvision/models/resnet.py. Accessed: 2024-12-18. 2023.

[45] 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.

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

[47] V7 Labs. *Vision Transformer Guide*. https://www.v7labs.com/blog/vision-transformer-guide. Accessed: 2024-12-18. 2024.

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

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

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

[51] 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.

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

[53] 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.

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

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

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

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

[58] 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.

[59] 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.

[60] 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. eprint: https://www.science.org/doi/pdf/10.1126/science.1127647. URL: https://www.science.org/doi/abs/10.1126/science.1127647.

[61] 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*. ICAIP '20. Chengdu, China: Association for Computing Machinery, 2021, pp. 42–47. ISBN: 9781450388368. DOI: 10.1145/3441250.3441264. URL: https://doi.org/10.1145/3441250.3441264.

[62] PyTorch Team. *MobileNetV2 — TorchVision 0.15.0 Documentation*. Accessed: 2024-12-22. 2024. URL: https://pytorch.org/vision/main/models/generated/torchvision.models.mobilenet_v2.html#torchvision.models.mobilenet_v2.

[63] TorchVision Contributors. *ConvNeXt Implementation in TorchVision*. https://github.com/pytorch/vision/blob/main/torchvision/models/convnext.py. Accessed: 2024-12-18. 2023.

[64] 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.

[65] 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*. Ed. by Abeba Birhane et al. Cham: Springer Nature Switzerland, 2025, pp. 3–20. ISBN: 978-3-031-64151-0.

[66] 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.

[67] 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.

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

[69] 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. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2024.123276. URL: https://www.sciencedirect.com/science/article/pii/S0957417424001416.

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

[71] Muhammad Shafiq and Zhaoquan Gu. "Deep Residual Learning for Image Recognition: A Survey". In: *Applied Sciences* 12.18 (2022). ISSN: 2076-3417. DOI: 10.3390/app12188972. URL: https://www.mdpi.com/2076-3417/12/18/8972.

[72] 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.

[73] 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.

[74] 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.

[75] 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.

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

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

[78] 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.

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

[80] PyTorch Te¡m. *StepLR*. https://pytorch.org/docs/stable/generated/torch.optim.lr_scheduler.StepLR.html. 2024.

[81] 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). ISSN: 2076-3417. DOI: 10.3390/app10103359. URL: https://www.mdpi.com/2076-3417/10/10/3359.

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

[83] PyTorch Contributors. *Adam*. https://pytorch.org/docs/stable/generated/torch.optim.Adam.html. 2024.

[84] 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.

[85] 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.

[86] 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. ISSN: 2307-387X. DOI: 10.1162/tacl_a_00675. URL: https://doi.org/10.1162/tacl%5C_a%5C_00675.

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

[88] 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.

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

Appendix A

Results

Tables of the results from training.

A.1 MobileNetV2

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

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

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

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

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

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

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

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

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

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

A.2 ResNet50V2

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

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

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

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

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

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

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

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

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

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

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

A.3 ViT-B/16

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

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

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

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

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

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

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

Loss Function	Balanced	Long-tailed	Head	Middle	Tail
Softmax loss	0.2254	0.4367	0.5071	0.1775	0.0263
Focal loss	0.2210	0.4206	0.4834	0.1953	0.0263
Weighted Softmax loss	0.1284	0.1760	0.1919	0.1302	0.0263
Class-balanced loss	0.0558	0.0076	0.0000	0.0237	0.1053
Balanced Softmax loss	0.2460	0.4244	0.4822	0.2130	0.0789
Equalization loss	0.2168	0.4215	0.4893	0.1716	0.0263
LDAM loss	0.1570	0.2750	0.3140	0.1361	0.0263

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

Loss Function	Balanced			Long-tailed			Head			Middle			Tail		
	Loss	Acc1	F1	Loss	Acc1	F1	Loss	Acc1	F1	Loss	Acc1	F1	Loss	Acc1	F1
Softmax	13.5272	0.2254	0.1871	6.7999	0.4367	0.4216	5.3024	0.5071	0.5248	11.3663	0.1775	0.2303	19.7511	0.0263	0.0263
Focal Loss	7.5701	0.2210	0.1850	3.6474	0.4206	0.4016	2.8064	0.4834	0.4914	6.1246	0.1953	0.2666	11.3091	0.0263	0.0263
Weighted Softmax	6.5391	0.1284	0.1144	3.9782	0.1760	0.1902	3.4559	0.1919	0.2357	4.7288	0.1302	0.1541	11.0975	0.0263	0.0351
Class-balanced	4.9938	0.0558	0.0368	5.8487	0.0076	0.0028	6.2065	0.0000	0.0000	4.7503	0.0237	0.0292	4.0694	0.1053	0.0746
Balanced Softmax	13.3583	0.2460	0.2123	6.7016	0.4244	0.4175	5.2929	0.4822	0.5121	11.3472	0.2130	0.2710	17.3287	0.0789	0.0877
Equalization	13.4511	0.2168	0.1786	6.7202	0.4215	0.4062	5.2340	0.4893	0.5051	11.6755	0.1716	0.2353	17.4650	0.0263	0.0263
LDAM	43.5990	0.1570	0.1321	17.1295	0.2750	0.2769	11.5903	0.3140	0.3466	30.5370	0.1361	0.1791	80.9656	0.0263	0.0351

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

A.4 ConvNeXt Base

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

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

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

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

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

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

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

Text.

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

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

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

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

Appendix B

Experimental Setup Details

B.1 Dataset Specifications

Table B.1 provides an overview of the dataset splits used in this thesis. The training set consists of 45,000 samples, with 450 samples per class, generated by splitting the original CIFAR-100 training set into training and test sets. The test set derived from the split consists of 5,000 samples, with 50 samples per class. The validation set, unaltered from the original CIFAR-100 test set, contains 10,000 samples with an equal distribution of 100 samples per class.

To simulate real-world class imbalance scenarios, an exponential imbalance was introduced into the training and test sets. The imbalance factor (imb_factor) was set to 0.01, resulting in a significant reduction of samples for the least frequent classes. Table B.2 provides a summary of the imbalance characteristics.

Additionally, the imbalanced test set was further divided into three subsets based on class frequencies in the training data:

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

The sample specifics are presented in table B.1.

Table B.1: Dataset Specifications

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

Table B.2: Imbalance Specifications

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

B.2 Data Preprocessing

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

Table B.3: Data Preprocessing Steps

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

B.3 Model Architecture Settings

Four different architectures were used: MobileNetV2, ResNet50V2, ViT-B/16, and ConvNeXt Base. All models were initialized with pretrained weights from ImageNet to leverage transfer learning. The modifications ensured that the architectures were adapted to the CIFAR-100 dataset while retaining the general features learned during pretraining.

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

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

ViT-B/16 The ViT-B/16 architecture is pretrained on the ImageNet-21K dataset and fine-tuned on the ImageNet-1K dataset [76]. The head is replaced with a 100-clas fully connected layer.

ConvNeXt Base The ConvNeXt Base architecture is pretrained on the ImageNet-1K dataset [79], and the final layer is replaced with a 100-class fully connected layer.

The specifications of the model architectures can be seen in table B.4 TODO: make this table prettier.

Table B.4: Model Architecture Settings

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

B.4 Training Configurations

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

- **Optimizer**: Adam
- **Learning Rate**: Initial value of `0.001`.
- **Learning Rate Scheduler**: StepLR with a step size of `30` epochs and a decay factor of `0.1`.
- **Batch Size**: `128`.
- **Number of Epochs**: `90`.
- **Weight Decay**: Default value of `1e-4`.
- **Class Weights**: Dynamically computed based on the training dataset and passed to the loss function. TODO: Reference to Methodology section.
- **Number of GPUs**: 4. See section B.6 for specifications.
- **Device Setup**: Utilized `torch.nn.DataParallel` for multi-GPU training.
- **Checkpoint Criteria**: The best model is saved based on the highest Top-1 validation accuracy.

B.5 Evaluation Metrics

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

B.6 Hardware and Software Configurations

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

B.6.1 Hardware

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

B.6.2 Software

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

B.7 Reproducibility Considerations

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

- **Random Seed**:
  - A fixed random seed of `42` was used for all experiments to ensure consistent initialization across runs.
  - Randomness was controlled for:

    - Python's `random` library.
    - NumPy (`np.random.seed`).
    - PyTorch (`torch.manual_seed` and `torch.cuda.manual_seed`).
  - `cudnn.deterministic` was set to `True` to enforce deterministic behavior in GPU computations.

- **Configuration Management**:
  - All hyperparameters, dataset settings, and model configurations were defined in YAML configuration files.
  - This allows for the exact replication of experiments by reusing the configuration files.

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

B.8 Implementation Faults

TODO: Revisit this section later on.

- The wrong version of the ViT-B/16 architecture might have implemented. The version implemented is the vit-base-patch16-224 [76] via timm which was pretrained on the ImageNet-21K and later fine-tuned on ImageNet-1K, whereas the other models are pretrained solely on the ImageNet-1K. The version on ViT-B/16 that is pretrained on the ImageNet-1K can be implemented as the torchvision version vit_b_16 [48].

- Class-Balanced Loss.