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

Master's Thesis in
Electrical Engineering

By

Christine Annelise Midtgaard
au521655
Study Program: Electrical Engineering

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

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

January 3, 2025

Abstract

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

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

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

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

Acknowledgements

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

Contents

Abstract **iii**

Acknowledgements **iv**

1 Introduction **1**
1.1 Problem Definition 1
1.1.1 Goals of this thesis 1
1.1.2 Approach 1
1.1.3 Scope of this thesis 2
1.2 Motivation 2
1.3 Reading Guide 2
1.4 Related Work 2

2 Background **3**
2.1 Long-Tailed Dataset 3
2.2 Model Architechtures 3
2.2.1 Convolutional Neural Networks 3
2.2.2 Visual Transformers 3
2.3 Classic Long-Tailed Methods 4
2.3.1 Class Re-balancing 4
2.3.2 Information Augmentation 6
2.3.3 Module Improvement 6

3 Methodology **7**
3.1 Overview of Methodology Approach 7
3.2 Algorithm Selection and Rationale 7
3.3 Long-tailed Learning Techniques 7
3.4 Loss Functions 7
3.5 Data Imbalance Handling Strategies 8
3.6 Evaluation Strategies 8
3.7 Implementation Details 8

4 Experimental Setup **9**
4.1 Dataset Specifications 9
4.2 Data Preprocessing 9
4.3 Model Architecture Settings 9

4.4 Training Configurations 9
4.5 Evaluation Metrics 9
4.6 Hardware and Software Configurations 10
4.7 Reproducibility Considerations 10

5 Results and Analysis **11**
5.1 Overall Results 11
5.2 Head, Middle, and Tail Class Performance 11
5.2.1 MobileNetV2 11
5.2.2 ResNet50V2 12
5.2.3 ViT-B/16 13
5.2.4 ConvNeXt Base 13
5.3 Comparison of Loss Functions 14
5.4 Qualitative Results 14
5.5 Summary and Discussion 15

6 Conclusion and Future Work **16**
6.1 Revisiting the Goals of the Thesis 16
6.2 Future Work 16

Bibliography **17**

A Results **18**
A.1 MobileNetV2 18
A.2 ResNet50V2 19
A.3 ViT-B/16 21
A.4 ConvNeXt Base 22

Chapter 1

Introduction

This thesis focuses on the problem with long-tailed datasets. The problem with training a deep learning model on long-tailed datasets is that the model will effectively the data from the classes with most samples, and not the classes with few samples. The finsihed model will then not recognize an input from the tail classes. Most real-world datasets follows a long-tailed structure, hence the need for a reliable method to detect examples of tail-class data. The aim of this thesis is to try out some of the methods tackling the long-tailed problem for deep learning described in the paper *Deep Long-Tailed Learning: A Survey* by Zhang et al.[1] to find a method for long-tailed learning that works on a specific long-tailed dataset of images of moths taken around equator. The goal of the moth dataset is to identify species.

1.1 Problem Definition

Define the problem formally, including key terms like "head classes" and "tail classes." Provide an example or visualization of a long-tailed dataset. Connect the problem definition to the moth dataset.

1.1.1 Goals of this thesis

Outline the goals of the thesis, emphasizing optimizing performance on tail classes. Mention how these goals contribute to the field of long-tailed learning.

Hypothesis

1.1.2 Approach

Summarize the approach to achieve goals, such as implementing and comparing methods from Zhang et al.'s survey.

1.1.3 Scope of this thesis

Specify the scope: Focus is on image classification. Methods are tested on a specific dataset. The evaluation is limited to certain metrics.

1.2 Motivation

Discuss why the problem is significant, including real-world implications. Mention the importance of biodiversity studies or the challenges of species identification with limited samples. Discuss broader impacts, such as how solving long-tailed learning problems can benefit other fields.

1.3 Reading Guide

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

1.4 Related Work

A section that describes the work related to this thesis.

Chapter 2

Background

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

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

2.1 Long-Tailed Dataset

A short introductory paragraph explaining why these topics are relevant and how they tie into the thesis. Contextualize the sections—e.g., "Long-tailed datasets are central to this thesis, as they represent the primary challenge. Model architectures and classic long-tailed methods form the foundation of the approaches explored in this work."

Mention the difference between class-imbalanced learning and long-tailed learning.

2.2 Model Architechtures

Describe the role of deep learning models in handling long-tailed datasets.

2.2.1 Convolutional Neural Networks

Add historical context. Mention specific CNNs used in this thesis (e.g., ResNet, MobileNet).

2.2.2 Visual Transformers

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

2.3 Classic Long-Tailed Methods

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

Following the paper *Deep Long-Tailed Learning: A Survey* [1], the existing deep long-tailed learning methods are grouped into three main categories based on their technical approach: class re-balancing, information augmentation, and module improvement. These categories are further divided onto sub-categories: re-sampling, class-sensitive learning, logit adjustment, transfer learning, data augmentation, representation learning, classifier desing, decoupled training, and ensemble learning (TODO: create a figure like figure 2 in the paper). This thesis does not aim to examine all the beforementioned method, but aims to find a deep learning approach to a specific problem. The backgrounds of the methods used in this thesis are described in this section.

2.3.1 Class Re-balancing

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

Re-sampling

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

Class-sensitive Learning

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

Loss Functions for Class-Sensitive Learning

The loss function serves as a measure of the model's fitness to the data, quantifying the distance between the actual and predicted values of the target. Typically, the

loss is represented as a nonnegative value, where smaller values indicate a better fit, and a perfect fit corresponds to a loss of zero [2].
Conventional training of deep networks using the softmax cross-entropy loss often overlooks class imbalance. This results in uneven gradients for different classes, leading to suboptimal performance on underrepresented classes. To mitigate this issue, modifications to the loss function are introduced to ensure a more balanced contribution from each class during training. One such technique is re-weighting which adjusts the training loss for different classes by assigning a specific weight to each class [1]. The softmax cross-entropy loss is used as a baseline, and is described below along with the loss functions for re-weighting.

Softmax Cross-Entropy Loss The *Softmax-Cross-Entropy loss*, often referred to as *softmax loss*, is a widely used combination for training deep neural networks in classification tasks, including image classification. It is particularly effective for multi-class problems, where the goal is to assign an input image to one of several predefined categories [3] [4].
The *Softmax* function transforms the raw output scores (logits) of the final layer of a neural network into a probability distribution over K classes. For an input $\mathbf{z} = [z_1, z_2, \ldots, z_K]$, the Softmax function for class i is defined as:

$$P(y = i \mid \mathbf{z}) = \frac{\exp(z_i)}{\sum_{j=1}^{K} \exp(z_j)} \tag{2.1}$$

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

$$\mathcal{L}_{\text{CE}} = -\sum_{i=1}^{K} y_i \log(P(y = i \mid \mathbf{z})) \tag{2.2}$$

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

$$\mathcal{L}_{\text{CE}} = -\log(P(y = c \mid \mathbf{z})) \tag{2.3}$$

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

Weighted Softmax Cross-Entropy Loss The *Weighted Softmax Cross-Entropy loss*, often referred to as *weighted softmax loss*, is a variant of the standard softmax cross-entropy loss, designed to address imbalanced datasets [4] [5]. By assigning different weights to each class, this method ensures that underrepresented classes contribute more to the overall loss, improving the model's performance on minority classes. The weighted cross-entropy loss applies class-specific weights to the standard cross-entropy formulation. It is defined as:

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

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

$$\mathcal{L}_{\text{WCE}} = -w_c \log(P(y = c \mid \mathbf{z})) \tag{2.5}$$

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

Focal Loss Focal Loss, introduced by Lin et al. (2017) [5], addresses the challenges of extreme class imbalance in classification tasks by dynamically scaling the standard cross-entropy loss. Focal Loss mitigates the issue of imbalanced datasets by down-weighting the loss contributions from well-classified examples and focusing on misclassified examples during training.

Class-Balanced Loss *Class-balanced loss*, introduced by Cui et al. (2019) [6], ...

Balanced Softmax Loss *Balanced Softmax loss*, introduced by Ren et al. (2020) [7], ...

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

Equalization Loss *Equalization loss*, introduced by Tan et al. (2020) [9], ...

2.3.2 Information Augmentation

Data augmentation techniques tailored for long-tailed datasets.

Transfer Learning

Data Augmentation

2.3.3 Module Improvement

Architectural changes to improve tail-class representation.

Chapter 3

Methodology

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

3.1 Overview of Methodology Approach

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

3.2 Algorithm Selection and Rationale

Description of the model architectures chosen, and why they are appropriate for long-tailed learning. Discussion of the strengths and limitations of these models in addressing the challenges posed by imbalanced data.

3.3 Long-tailed Learning Techniques

Description of the specific methods used to address class imbalance, such as data sampling, class re-weighting, etc. Justification for selecting these techniques, potentially referencing prior research (from Deep Long-Tailed Learning: A Survey by Zhang et al.).

3.4 Loss Functions

Explanation of the different loss functions explored, such as cross-entropy, focal loss, LDAM loss, etc., and their relevance for long-tailed learning. Rationale for each loss function's inclusion, focusing on its expected benefits for imbalanced classes and how it addresses the bias toward majority classes.

3.5 Data Imbalance Handling Strategies

Detailed explanation of the techniques for creating and handling an imbalanced dataset, such as generating imbalanced training and test sets.

3.6 Evaluation Strategies

Justification for the metrics and evaluation approach, such as using weighted or macro F1 scores. Explanation of how the performance is assessed across different class groups to capture the model's performance on minority classes.

3.7 Implementation Details

Technical explanations of any unique or customized methods implemented in code, for example the custom dataset.

Chapter 4

Experimental Setup

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

4.1 Dataset Specifications

Details about the dataset(s) used, including size, source, and preprocessing steps. Description of class imbalance characteristics and the train/validation/test splits.

4.2 Data Preprocessing

Any transformations, augmentations, or normalization applied to the dataset before feeding it to the model. Information on how the class imbalance is handled (re-sampling or synthetic data generation).

4.3 Model Architecture Settings

Description of the models used, including any specific architecture choices, hyperparameters, or modifications. Brief details on why these models were chosen.

4.4 Training Configurations

Hyperparameters, such as batch size, learning rate, optimizer type, and regularization techniques (dropout, weight decay). Any specific settings for handling long-tailed data, such as DRW.

4.5 Evaluation Metrics

Explanation of the metrics used to assess model performance. Justification for choosing each metric.

4.6 Hardware and Software Configurations

Hardware details. Software environment, including the versions of libraries and frameworks.

4.7 Reproducibility Considerations

Steps taken to ensure that results can be reproduced, such as random seed initialization and details on dataset versions. Scripts, configurations, or instructions for reproducing experiments.

Chapter 5

Results and Analysis

Presentation of the results, with tables, charts, and explanations for each tested method's performance.

Brief overview of the chapter's purpose. Recap the evaluation goals (model performance across head, middle, and tail classes, and comparing methods).

5.1 Overall Results

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

5.2 Head, Middle, and Tail Class Performance

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

5.2.1 MobileNetV2

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

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

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

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

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

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

5.2.2 ResNet50V2

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

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

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

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

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

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

5.2.3 ViT-B/16

Table 5.5 show the top 1 accuracies for ViT-B/16 on various loss functions.

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

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

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

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

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

5.2.4 ConvNeXt Base

Table 5.7 show the top 1 accuracies for ConvNeXt Base on various loss functions.

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

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

Table 5.8 show the top 1 accuracies for ConvNeXt Base on various loss functions.

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

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

5.3 Comparison of Loss Functions

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

5.4 Qualitative Results

Optional.

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

5.5 Summary and Discussion

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

Chapter 6

Conclusion and Future Work

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

6.1 Revisiting the Goals of the Thesis

6.2 Future Work

Bibliography

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

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

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

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

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

[6] 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.

[7] 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.

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

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

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.

Text.

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

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

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

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

A.4 ConvNeXt Base

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

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

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

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

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	data	data	data	data	data	data	data	data	data	data	data	data	data	data	data

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	data	data	data	data	data

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	data	data	data	data	data	data	data	data	data	data	data	data	data	data	data

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