LT-Soups: Bridging Head and Tail Classes via Subsampled Model Soups

Masih Aminbeidokhti[1,*] **Subhankar Roy**[2]

Eric Granger[1] **Elisa Ricci**[3,4] **Marco Pedersoli**[1]

[1]École de technologie supérieure, [2]University of Bergamo
[3]University of Trento, [4]Fondazione Bruno Kessler (FBK)

Abstract

Real-world datasets typically exhibit long-tailed (LT) distributions, where a few head classes dominate, and many tail classes are severely underrepresented. While recent work shows that parameter-efficient fine-tuning (PEFT) methods like LoRA and AdaptFormer preserve tail-class performance on foundation models such as CLIP, we find that they do so at the cost of head-class accuracy. We identify the head-tail ratio, the proportion of head to tail classes, as a crucial but overlooked factor influencing this trade-off. Through controlled experiments on CIFAR100 with varying imbalance ratio (ρ) and head-tail ratio (η), we show that PEFT excels in tail-heavy scenarios but degrades in more balanced and head-heavy distributions. To overcome these limitations, we propose LT-Soups, a two-stage model soups framework designed to generalize across diverse LT regimes. In the first stage, LT-Soups averages models fine-tuned on balanced subsets to reduce head-class bias; in the second, it fine-tunes only the classifier on the full dataset to restore head-class accuracy. Experiments across six benchmark datasets show that LT-Soups achieves superior trade-offs compared to both PEFT and traditional model soups across a wide range of imbalance regimes.

1 Introduction

In machine learning, balanced class distributions are often assumed in both theory and practice [12, 73, 28]. However, real-world datasets frequently deviate from this assumption, exhibiting severe class imbalance where a few head classes dominate and tail classes remain significantly underrepresented [56, 19, 34]. This imbalance poses a fundamental challenge: models must learn effectively from limited tail-class data while preserving overall robustness [9].

Recent advances in vision-language foundation models, particularly CLIP [43], have introduced promising tools for addressing class imbalance. Trained on large-scale, diverse datasets, CLIP demonstrates strong robustness to distributional shifts and has become a popular backbone for long-tailed recognition [59, 58, 37, 55, 35]. Building on this, Shi et al. [50] achieve state-of-the-art results by applying parameter-efficient fine-tuning (PEFT) methods such as Low-Rank Adaptation (LoRA) [20] and AdaptFormer [8], in combination with logit adjustment (LA) loss [40, 46], which incorporates class priors by adding a class-dependent offset to the logits. While this PEFT-based approach improves overall and tail-class accuracy, they observe that it still underperforms full fine-tuning in certain regimes.

*Correspondence to: masih.aminbeidokhti.1@ens.etsmtl.ca. Code at https://github.com/Masseeh/LT-Soups.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

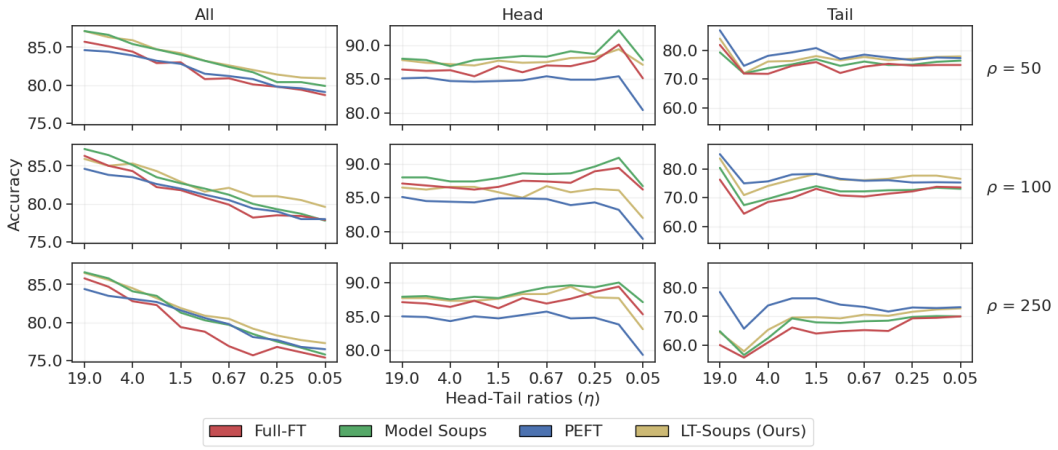


Figure 1: Performance of baselines and LT-Soups on the CIFAR100 benchmark varying ρ and η. While full fine-tuning generally outperforms PEFT on head classes, PEFT demonstrates superior performance on tail classes. In contrast, our approach maintains robust accuracy across all imbalance settings, showing resilience to shifts in both the sample distribution and class structure.

These observations motivate a deeper investigation into when and why PEFT is effective. To this end, we construct a controllable long-tailed benchmark based on CIFAR100 that allows systematic variation in both the sample counts across classes, quantified by the imbalance ratio (ρ), and the number of classes above or below a sample threshold, defined as the head-tail ratio (η). This setup enables a more fine-grained analysis of imbalance structures. Within this framework, we compare two full fine-tuning strategies against a PEFT baseline: full fine-tuning with logit adjustment (LA) and model soups [60], which averages the weights of multiple LA-trained models initialized with different seeds and hyperparameters. Results from this benchmark confirm previous findings: on average, PEFT improves overall accuracy (80.8 vs. 81.2) and tail-class accuracy (76.4 vs. 70.2) compared to full fine-tuning, but at the cost of degraded head-class performance (87.0 vs. 84.3). A detailed breakdown in Figure 1 shows that PEFT is especially effective in tail-heavy scenarios, where rare classes dominate ($\eta \ll 1$), but its performance declines as the class structure becomes more balanced or head-heavy, highlighting its limited robustness to shifts in class structure.

This highlights a key trade-off: PEFT helps prevent overfitting and supports tail classes, but lacks adaptability in more balanced and head-heavy settings. Conversely, full fine-tuning offers stronger adaptation but requires careful regularization. Model soups offer a middle ground by averaging models trained with different seeds and hyperparameters [29, 27], but our experiments show that traditional soups, built from models trained on the same imbalanced dataset, still underperform in tail-heavy cases, as they tend to overemphasize head-class performance due to the dominance of high imbalance ratios.

To address these limitations, we introduce LT-Soups, a two-stage model soups framework designed to deliver robust performance across diverse imbalance scenarios, jointly characterized by the imbalance ratio (ρ) and the head-tail ratio (η). Unlike PEFT, which performs well primarily in tail-heavy settings, LT-Soups consistently achieves strong results across tail-heavy, balanced, and head-heavy class structures (Figure 1). In the *first* stage, LT-Soups constructs a weight ensemble by averaging multiple fully fine-tuned models, each trained on a subset exhibiting a distinct imbalance ratio, collectively spanning a spectrum of imbalance ratios. The aim is to create models that "specialize" for each of these imbalance ratios, when averaged, promote a balanced representation that performs well on both the head and the tail classes. To recover any head-class information lost during subsampling, the *second* stage fine-tunes only the classifier on the full dataset using class-balancing techniques. By seamlessly combining the adaptability of full-rank optimization, favouring the head-classes, and the robustness of weight ensembling for the tail-classes, LT-Soups strikes a better trade-off than PEFT and model soups, and thus bridges the head and the tail classes.

Our contributions are threefold: **(1)** We introduce a dual-axis framework for characterizing class imbalance using both the imbalance ratio (ρ) and head-tail ratio (η), and show how they jointly affect performance. **(2)** We propose **LT-Soups**, a novel two-stage approach that mitigates representation

bias and adapts effectively across a broad range of imbalance structures. **(3)** We conduct extensive experiments on five benchmark datasets and show that while existing LT methods perform well only under specific imbalance configurations, our approach consistently delivers robust, all-around performance across a wide range of imbalance scenarios.

2 Related Work

Imbalanced Learning. Class imbalance has traditionally been tackled through oversampling minority classes, undersampling majority classes, or applying reweighted loss functions such as focal loss or logit adjustment (LA) [7, 17, 40]. While effective in certain settings, these techniques often struggle under overparameterized models [66]. Decoupled training frameworks further refine this process by separating representation learning and classifier training [25, 67], assuming biases lie primarily in the classifier layer. However, this assumption breaks down when adapting foundation models, as full fine-tuning can lead to catastrophic forgetting and degraded generalization for both head and tail classes [41, 50].

Ensemble-based methods address class imbalance by combining experts trained on diverse data distributions [4, 53]. Examples include BBN [74] and RIDE [57], which use architectural branching or dynamic routing, and LFME [62], which employs group-wise distillation. Mixture-of-Experts approaches such as SADE [68], Mdcs [69], and DirMixE [65] merge experts trained with different logit adjustments (e.g., uniform, long-tail, inverse long-tail). Unlike these methods that require all experts at inference, LT-Soups collapses multiple fine-tuned models into a single network via weight averaging, offering an inference-efficient alternative. While prior works rely on specialized architectures and heuristic expert definitions, our approach retains architectural simplicity by using parallel fine-tuning on controlled subsamples and model averaging to preserve both the generalization and efficiency of the foundation model.

CLIP and other vision-language models exhibit inherent robustness to class imbalance, largely due to the diversity of their pretraining data [59]. This robustness has been further extended through techniques such as prompt tuning [13], retrieval-based augmentation [35], and joint vision-language training paradigms [37, 58]. While these methods improve adaptation to long-tailed distributions, Shi et al. [50] show that PEFT combined with logit adjustment (LA) loss achieves state-of-the-art performance by selectively adapting CLIP's pretrained features. However, they also observe that this comes at the cost of reduced head-class accuracy.

In this work, we demonstrate that PEFT is particularly effective in tail-heavy scenarios, but its performance diminishes as the class structure becomes more balanced or skews toward head-class dominance. To overcome this limitation, we propose a method designed to maintain robust performance across the entire long-tail distribution spectrum. Our approach merges models trained on a subset exhibiting a distinct imbalance ratio, collectively spanning a spectrum of imbalance ratios, enabling the final model to achieve balanced accuracy across both head and tail classes.

Model Merging. Methods based on model merging, or weight averaging, has emerged as a practical strategy for reducing communication overhead in federated and distributed settings [39, 15], improving robustness to distribution shifts [60], and enhancing generalization through techniques like SWA and EMA [23, 54]. Recent efforts also apply merging for continual learning and RLHF fine-tuning [1, 45]. Yet, to our knowledge, model merging has not been explored for imbalanced classification.

3 A Closer Look at Imbalanced Learning with Foundation Models

3.1 Preliminaries

Given training data $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$, where $\boldsymbol{x}_i$ denotes an input sample and $y_i \in \mathcal{C}$ is its corresponding class label from a set of $K = |\mathcal{C}|$ classes. Let n_j denote the number of training samples for class j, and let the total number of training samples be $N = \sum_{j=1}^K n_j$. Without loss of generality, we assume that classes are sorted in decreasing order of frequency, *i.e.*, if $i < j$, then $n_i \geq n_j$. In the imbalanced setting considered here, the most frequent class is significantly larger than the rarest one, such that $n_1 \gg n_K$. To quantify this imbalance, we define the imbalance ratio as

$\rho = n_K/n_1$. Following [34], we categorize classes with more than 100 training samples ($n_j > 100$) as *head* classes, and the rest as *tail* classes.[2] Since we aim to achieve balanced performance across all classes, we report BalAcc $= \frac{1}{|\mathcal{C}|}\sum_{c\in\mathcal{C}}$ Accuracy(c) which equally weights performance on each class.

Our model is composed of two main components: a feature extractor and a classification head. For feature extraction, we adopt the CLIP vision encoder, implemented using a Vision Transformer (ViT) [14] and parameterized by θ. The representation for input $\boldsymbol{x}$ is given by $f_I(\boldsymbol{x};\theta) = \boldsymbol{z}$ where $\boldsymbol{z}$ is the extracted feature vector. The final class prediction is computed as $\hat{y} = \arg\max g(\boldsymbol{z};\omega)$ where g denotes the prototypical classification head with parameters ω constructed from the CLIP text encoder. (see Appendix D for the full details).

Previous work suggests that training with standard *Cross-Entropy* loss with instance-balanced sampling often leads to head-class bias due to class imbalance [5, 17]. *Logit Adjustment* (LA) [40] addresses this by adding a class-dependent offset to the logits, thereby correcting for prior class frequencies as follows:

$$\ell_{LA}(y, g(\boldsymbol{z})) = -\log \frac{\exp(g_y(\boldsymbol{z}) + \log \pi_y)}{\sum_{y'\in\mathcal{C}} \exp(g_{y'}(\boldsymbol{z}) + \log \pi_{y'})} \tag{1}$$

where g_y denotes the predictive logit of model on class y and $\pi \in \Delta_y$ are estimates of the class priors $\mathbb{P}(y)$ based on the empirical class frequencies on the training data D. However, Shi et al. [50] observed that when fine-tuning CLIP models from pretrained weights using LA (referred to as *Full-FT*), the resulting class-conditional distributions can become inconsistent, particularly for tail classes. To mitigate this, they advocate for methods that preserve proximity to the pretrained initialization, leveraging PEFT strategies such as LoRA and AdaptFormer.

3.2 Characterizing Imbalanced Distribution with Head-Tail Ratio

In practice, class imbalance can manifest in diverse structural forms. While the imbalance ratio (ρ) is a standard metric for quantifying distributional skew, we show that it is insufficient to fully capture the complexity of long-tailed distributions. As a complementary measure, we introduce the head-tail ratio (η), which reflects the relative number of head versus tail classes and emphasizes the underlying class structure.

Definition 1. Let $\mathcal{H} = \{c \mid n_c > \tau\}$ and $\mathcal{T} = \{c \mid n_c \leq \tau\}$ denote the sets of head and tail classes, respectively, based on a sample threshold τ. Let $H = |\mathcal{H}|$ and $T = |\mathcal{T}|$ be the number of head and tail classes. The head-tail ratio is then defined as $\eta = \frac{H}{T}$.

To investigate the joint effect of ρ and η on model performance, we construct a synthetic benchmark based on CIFAR100, where both parameters are systematically varied. For a fixed η, classes are partitioned into head and tail groups, and within each group, sample sizes are assigned following an exponential decay distribution. This procedure is repeated across 11 values of η, ranging from 19 (head-heavy) to 0.05 (tail-heavy), and for $\rho \in \{50, 100, 250\}$. In these configurations, head-class sample sizes range from 500 to 101, and tail-class sizes from 100 to 2 (see Figure 7a in the Appendix for visualization).

Figure 2 presents performance trends marginalized over varying ρ, η, and their joint effects. The results reveal that no single method consistently dominates; instead, the best-performing approach shifts depending on the imbalance configuration. In tail-heavy regimes (low η), PEFT methods excel due to their ability to retain generalizable pretrained features for underrepresented classes. Conversely, in head-heavy settings (high η), full fine-tuning becomes more advantageous, leveraging its flexibility to fit the dominant head-class structure. These trends underscore the need for methods that can adapt effectively across the full spectrum of imbalance scenarios.

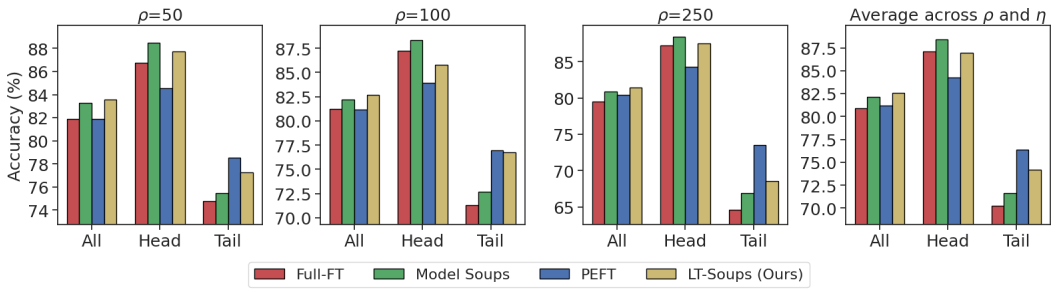
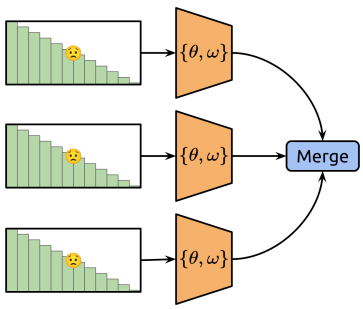
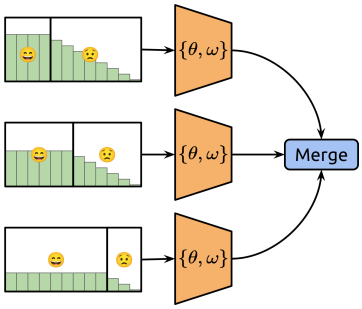


Figure 2: Marginalized performance of baselines, including LT-Soups, on CIFAR100 across varying ρ and η. The first three columns average over η for each ρ; the last column averages over all configurations. Refer to Figure 1 for the detailed results.



(a) Each model within Model Soups fine-tunes the full training set.



(b) Each model within LT-Soups fine-tunes a subset of less imbalanced data from the full training set.

Figure 3: Comparison between Model Soups and LT-Soups. (a) Model Soups merges models fine-tuned on full, severely imbalanced training data. (b) LT-Soups merges models fine-tuned on subsets with increasingly higher imbalance ratios to preserve pretrained features while adapting to class distribution shifts.

4 LT-Soups: Imbalanced Learning by Subsampled Model Averaging

The preceding toy experiment illustrates that the optimal method depends on the underlying class structure. In head-heavy distributions, full fine-tuning is particularly effective, as it adjusts all model parameters to capture the rich structure of frequent classes. In contrast, when tail classes dominate, PEFT approaches like LoRA and AdaptFormer (as used in LIFT [50]) perform better by preserving pretrained representations that generalize well under limited supervision. Motivated by this trade-off, our goal is to design a method that maintains balanced performance across both extremes, regardless of the imbalance pattern.

Ensemble methods such as model soups [60] (Figure 3a) have demonstrated effectiveness in improving both overall and minority-class performance [29, 27]. However, as shown in the previous section, traditional soups, while outperforming single-model fine-tuning, remain suboptimal for the full spectrum of long-tailed distributions. This is especially true in tail-heavy scenarios, where they tend to overemphasize head-class performance due to the dominance of high imbalance ratios in the training data. We address this limitation with LT-Soups, a soups-based framework specifically designed for long-tailed distributions. Each model in LT-Soups is fine-tuned on a subset of the training data with a distinct, reduced imbalance ratio (Figure 3b). While such subsampling enhances tail-class learning [6], it may omit valuable head-class information [26]. To balance this, we train models on subsets with gradually increasing imbalance levels, allowing each model to specialize in different regions of the long-tail spectrum. The final model is obtained by averaging these specialized models. To further recover any lost head-class information, we introduce a second stage where only the classifier head is fine-tuned on the full dataset using a class-balanced objective (e.g., LA loss).

[2]For simplicity, we initially group all low-resource classes into a single tail category. In the experimental section, we further subdivide the tail into *medium-shot* and *few-shot* groups for consistency with prior work.

Algorithm 1 LT-Soups (Parallelizable Pseudocode)

Input: θ_0 pre-trained weights, full training data D, $\{D_{\rho_n}\}_{n=1}^{MN}$ subsets with N imbalance ratios ρ_n and M bootstrapping per ρ_n, λ merging interpolation.
Training: **for all** $n = 1$ to MN **in parallel do**
    $\theta_n \leftarrow \text{FineTune}(\theta_0, D_{\rho_n})$
Prepare models: Sort models in an increasing order of ρ_n
Weight Averaging: $\forall n = 1$ to N, $\theta_n = (1-\lambda)\theta_n + \lambda\theta_{n-1}$
Re-train final classifier on full D

Pseudocode for LT-Soups is provided in Algorithm 1, and the remainder of this section details the full procedure.

Balanced representation. Subsampling is a common strategy for addressing class imbalance by reducing overrepresented head-class samples [17, 6]. However, aggressive subsampling can discard useful head-class information, degrading overall performance [26, 7, 49]. To address this, we propose *progressive subsampling*, which incrementally increases the imbalance ratio across subsets. Each model is fine-tuned on a subset with a specific ratio, preserving tail-class data while managing head-class underutilization. We construct the subset sequence as $\left\{D_{\rho_i} \;\middle|\; \rho_i = 2^i,\ i \in \{0, 1, 2, \ldots, \lceil \log_2(\rho) \rceil\}\right\}$ where D_{ρ_i} is a dataset with imbalance ratio ρ_i. This yields a sparse sequence of subsets with exponentially increasing imbalance ratios, ensuring broad coverage while limiting the number of models in the soups procedure. In practice, we retain only the first N subsets, as extremely high imbalance ratios tend to overly favor head classes and degrade tail-class performance.

The resulting N models, each trained on a different subset, are merged using a recursive interpolation strategy. Given weights $\{\theta_n\}_{n=0}^{N}$, where θ_0 is the pretrained model, LT-Soups recursively combines models via $\theta_n = (1-\lambda)\theta_n + \lambda\theta_{n-1}$. The interpolation coefficient λ controls knowledge retention from previous stages. This procedure ensures (1) proximity to the pretrained model θ_0, preserving CLIP's zero-shot capabilities [61], and (2) smooth integration of head and tail class representations [74]. As shown in Section 5.3, LT-Soups exhibits partial insensitivity to the choice of loss function, owing to the balance introduced by subsampling and model averaging. However, since each subset remains mildly imbalanced, albeit less so than the full training set, applying LA loss during fine-tuning further mitigates the effects of label distribution shifts.

Variance reduction. While weight averaging and subsampling help mitigate head-class dominance, fine-tuning large pretrained models can still lead to degradation in tail-class performance [59]. To address this, we maintain an exponential moving average (EMA) of model weights via $\theta_{ema} = (1-\mu)\cdot\theta_{ema} + \mu\cdot\theta$, with a momentum coefficient $\mu = 0.99$. EMA acts as a regularizer during training [21], promoting convergence to flatter minima [23], which has been shown to enhance generalization, particularly for underrepresented classes.

Since subsampling reduces data per subset and introduces variance, we adopt a bootstrapping strategy inspired by bagging [2]: for each subset, we train M models on different bootstrap samples and uniformly average their weights. This stabilizes learning and yields more robust representations.

Classifier re-training. To further recover head-class information lost during subsampling, we perform a final fine-tuning stage on the classifier head using the full training set. The backbone is frozen to preserve merged representations, and LA loss is applied to adjust decision boundaries based on label frequencies. This step improves head-class accuracy without harming tail-class performance, similar to calibration in two-stage LT methods [25].

5 Experiments

5.1 Datasets and evaluation protocol.

We evaluate our method on both synthetically constructed and naturally occurring long-tailed (LT) datasets. For synthetic benchmarks, we use CIFAR-100-LT, ImageNet-LT, and Places-LT—long-tailed variants derived from their balanced counterparts by sampling class instances according

to Pareto or exponential decay distributions [34]. These datasets exhibit sample counts ranging from 1,280 to as few as 5 images per class. For real-world evaluation, we include iNaturalist 2018 (8,142 classes, 437.5K images) and NIH-CXR-LT (20 classes, 88.5K images), which reflect different imbalance structures, with approximately 10% and 90% head classes, respectively. To assess performance across the long-tail spectrum, we also report the average accuracy across all five datasets. Following [34], we evaluate separately on many-shot (>100 samples), medium-shot (20–100), and few-shot (<20) class subsets. For ablation analysis, we use TinyImageNet-LT, which contains 200 classes with sample counts ranging from 500 in head classes to 5 in tail classes. To conserve space, we present only CLIP-based results in the main text; additional implementation details and extended results are included in Appendix D.

Table 1: Comparison with state-of-the-art methods on synthetic LT distributions.

Methods	CIFAR100-LT ρ=100 η=0.54				Places-LT ρ=996 η=0.55				ImageNet-LT ρ=256 η=0.62			
	All	Many	Med.	Few	All	Many	Med.	Few	All	Many	Med.	Few
BALLAD [37]	-	-	-	-	49.5	49.3	50.2	48.4	75.7	79.1	<u>74.5</u>	69.8
Decoder [58]	-	-	-	-	46.8	-	-	-	73.2	-	-	-
LPT [13]	-	-	-	-	50.1	49.3	52.3	46.9	-	-	-	-
Linear Probing	70.0	77.2	71.1	60.4	48.8	48.8	49.7	47.1	74.2	77.8	73.3	67.4
Full-FT	79.6	88.1	79.9	69.3	46.6	49.9	46.3	41.4	73.9	79.8	71.9	63.9
cRT [25]	78.8	<u>89.7</u>	79.7	65.1	44.4	51.0	43.1	35.4	72.6	81.1	70.6	56.1
PEFT [50]	81.3	85.2	80.9	<u>77.1</u>	<u>51.5</u>	<u>51.3</u>	<u>52.2</u>	**50.5**	<u>77.0</u>	80.2	**76.1**	**71.5**
Model Soups [60]	<u>82.1</u>	**89.9**	<u>82.2</u>	73.0	49.4	**51.7**	50.0	43.7	76.0	**81.5**	<u>74.5</u>	65.5
LT-Soups (Ours)	**83.5**	88.2	**83.5**	**78.0**	**51.7**	51.2	**52.8**	<u>50.3</u>	**77.4**	<u>81.2</u>	**76.1**	<u>70.7</u>

Table 2: Comparison with state-of-the-art methods on real-world LT distributions.

Methods	NIH-CXR-LT ρ=6491 η=5.66				iNaturalist 2018 ρ=500 η=0.11			
	All	Many	Med.	Few	All	Many	Med.	Few
BALLAD [37]	34.5	36.7	38.9	20.8	49.5	49.3	50.2	48.4
Decoder [58]	-	-	-	-	59.2	-	-	-
LPT [12]	-	-	-	-	76.1	-	-	79.3
Linear Probing	17.5	13.3	21.1	16.7	60.4	48.9	60.0	63.9
Full-FT	38.0	<u>43.8</u>	**41.5**	20.0	76.1	75.7	76.9	75.3
cRT [25]	37.7	42.9	39.3	25.0	44.4	51.0	43.1	35.4
PEFT [50]	<u>38.5</u>	43.3	40.4	<u>25.5</u>	**79.1**	72.4	**79.0**	**81.1**
Model Soups [60]	38.0	**45.6**	40.2	20.0	76.4	**77.1**	76.8	75.6
LT-Soups (Ours)	**39.3**	42.4	<u>40.7</u>	**30.9**	<u>78.2</u>	76.7	<u>78.5</u>	<u>78.2</u>

5.2 Main results

Synthetic LT datasets. Table 1 presents the accuracy of LT-Soups on three benchmark datasets with synthetically induced long-tail distributions: CIFAR100-LT, Places-LT, and ImageNet-LT. Our method outperforms all state-of-the-art baselines in overall accuracy on every dataset. Notably, LT-Soups surpasses Model Soups and PEFT, the most competitive baselines. While PEFT achieves competitive performance on tail classes through low-rank adaptation, it often does so at the cost of many-shot accuracy, especially in CIFAR100-LT, where LT-Soups maintains strong tail accuracy (78.0) without sacrificing performance on many-shot (88.2) or medium-shot (83.5) categories. Model Soups, on the other hand, tends to overfit many-shot categories (e.g., 89.9 on CIFAR100-LT) but underperforms on few-shot classes due to averaging independently fine-tuned models without accounting for class imbalance.

Real-world LT datasets. In Table 2, we evaluate LT-Soups on two naturally imbalanced datasets—NIH-CXR-LT and iNaturalist 2018—which present distinct challenges. (1) NIH-CXR-LT consists primarily of head-class (many-shot) samples but diverges significantly from CLIP's pretraining domain, as it comprises medical X-ray images. (2) iNaturalist 2018 is heavily skewed toward medium- and few-shot categories and is more closely aligned with CLIP's natural image priors. On NIH-CXR-LT, LT-Soups achieves the highest overall accuracy (39.3%), outperforming PEFT (38.5%) and delivering substantial gains in the few-shot regime (+5 points over PEFT and +10 over

Model Soups). On iNaturalist, where PEFT performs strongly (79.1% overall), LT-Soups remains competitive (78.2%) while offering more balanced accuracy across many-, medium-, and few-shot subsets.

Full spectrum results. The datasets in our benchmark exhibit diverse long-tail characteristics, with imbalance ratios ranging from 100 to 6,491 and head-tail ratios spanning 0.11 to 5.66. To abstract away the effects of individual dataset characteristics, Figure 4 reports the average accuracy of Full-FT, Model Soups, PEFT, and LT-Soups across all five benchmarks. While PEFT performs well on medium-shot and few-shot splits, and Model Soups excels on many-shot classes, LT-Soups consistently achieves strong performance across all splits, demonstrating its robustness across the long-tail spectrum.

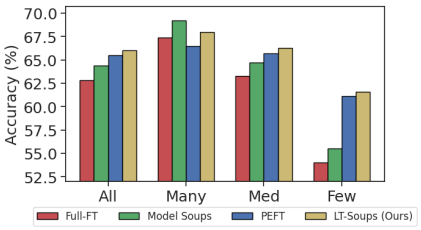


Figure 4: Average performance across 5 LT benchmarks.

5.3 Empirical analysis of LT-Soups

In this section, we provide a comprehensive analysis of LT-Soups from multiple perspectives (due to space constraints, some of the analysis is provided in the Appendix A).

Effect of subsampling. LT-Soups averages the weights of NM models, where N is the number of subsets used during each fine-tuning and M is the number of bootstraps per subset. For TinyImageNet-LT (imbalance ratio 100 and head-tail ratio 0.3), we use N=8 and M=2. To show the impact of our proposed weight averaging scheme, we compare this with Soups-ρ_n baselines that follow the same two-stage framework as LT-Soups, except in the first stage, they average 16 models, all trained on subsets with the same imbalance ratio ρ_n. Notably, Soups-100 aligns with the traditional model soup approach [60], where the weights of 16 fully fine-tuned models on the entire dataset are averaged. As shown in Table 3, all of the soups baselines consistently outperform full fine-tuning, regardless of the subset choice. However, results show that different imbalance ratios yield varying outcomes across head and tail categories. For example, Soups-8 achieves the highest tail accuracy of 75.0, whereas Soups-100 reaches the highest head accuracy of 85.9. Rather than optimizing for a single imbalance ratio, LT-Soups applies weight averaging across the full spectrum, effectively merging the advantages of both approaches to achieve a more balanced overall trade-off. (See Table 10 in the Appendix for a similar analysis on PEFT.)

Table 3: Comparison of LT-Soups and Soups-ρ each with a total of 16 models across All, Head, and Tail accuracy.

	Full-FT	PEFT	Soups-1	Soups-2	Soups-4	Soups-8	Soups-16	Soups-32	Soups-64	Soups-100	LT-Soups
All	73.2	77.1	71.7	75.9	76.0	77.2	77.2	77.3	77.9	77.6	78.6
Head	83.4	83.0	74.6	78.6	78.7	81.0	82.8	84.7	85.5	85.9	85.0
Tail	67.7	73.9	70.1	74.4	74.6	75.0	74.1	73.3	73.7	73.0	75.2

Effect of classifier re-training (CR). We found that additional final-layer tuning with logit adjustment on PEFT and Model Soups has little to no effect. Table 4 summarizes the results on TinyImageNet-LT. We hypothesize that, unlike these baselines, LT-Soups does not fully exploit the entire training set, due to the downweighting effect introduced by weight averaging. Consequently, fine-tuning the final layer helps LT-Soups recover head-class sharpness and improves overall performance.

Component analysis. LT-Soups is designed to balance effective task adaptation with minimal deviation from pretrained weights. Figure 5 shows the cumulative effect of its components on accuracy and weight change. Starting from Full Fine-Tuning, which causes the largest deviation from the CLIP zero-shot model (35.4), each component incrementally improves performance while reducing or controlling weight deviation. EMA offers a modest accuracy boost with minimal impact on weight shift. Subsampling and model merging significantly improve tail accuracy (+6.3) and reduce weight change to 12.7, highlighting the benefit of balanced training. Bootstrapping stabilizes training further, slightly improving head accuracy. Classifier re-training refines decision boundaries,

Table 4: Comparison of baseline methods including LT-Soups with and without classifier re-training (CR).

Method	All	Head	Tail
PEFT	77.1	83.0	73.9
PEFT + CR	77.0	83.0	73.8
Model Soups	77.6	85.9	73.0
Model Soups + CR	77.6	85.5	73.4
LT-Soups Stage 1	78.1	84.9	74.5
LT-Soups	**78.6**	**85.0**	**75.2**

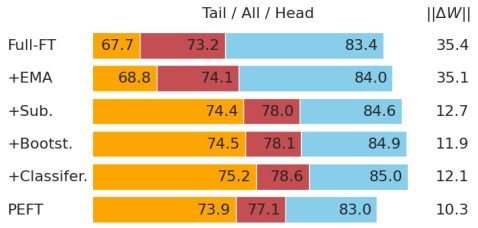


Figure 5: Performance and weight change comparison across different stages of LT-Soups.

Table 5: Comparison of our merging strategy with uniform merging across two datasets exhibiting distinct long-tailed distributions.

	TinyImageNet-LT		iNaturalist 2018	
	Ours	Unifrom	Ours	Unifrom
All	78.6	78.5	78.2	74.7
Many	85.0	83.4	76.7	67.4
Med.	78.3	78.4	78.5	75.8
Few	71.5	72.9	78.2	75.3

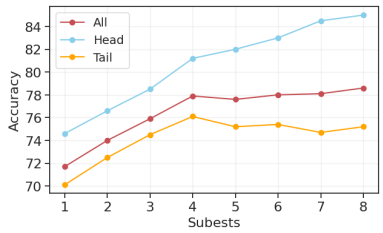


Figure 6: Performance across number of subsets, N, each with increasing imbalance ratios on TinyImageNet-LT.

yielding the highest overall and head accuracy. Compared to PEFT, LT-Soups shows a slightly higher weight change (12.1 vs. 10.3) but delivers better accuracy across all class groups. This reflects its ability to adapt meaningfully while preserving pretrained knowledge.

Number of subsets. Figure 6 illustrates the impact of the number of subsets N, each with increasing imbalance ratios, used in LT-Soups during fine-tuning. In this experiment, the interpolation weight λ and M are fixed at 0.7 and 2, respectively. As N increases, head-class accuracy steadily improves—from 74.6 at N=1 to 85.0 at N=8—while tail-class accuracy peaks at 76.1 when N=3. The best overall trade-off is observed at N=8, indicating it as the most balanced configuration.

Merging strategies. LT-Soups recursively merges models trained on subsets with progressively higher imbalance ratios. One intuitive way to think about this merging procedure is to interpret it as an exponential moving average (EMA) over fine-tuned models sorted by increasing imbalance severity, with a tunable parameter that adjusts the influence of more balanced (but smaller) versus less balanced (but larger) subsets. In this section, we compare this strategy against uniform WA, which applies a simple arithmetic mean, giving equal weight to all models regardless of their imbalance level.

Table 5 confirms our hypotheses. In particular, we compare recursive WA and uniform WA across two datasets with different similarities compared to CLIP-pretrained weights (according to the zero-shot performance). On TinyImageNet-LT, which is already well-aligned with CLIP-pretrained features, there is little to no difference between the two averaging schemes. However, for datasets that require significant adaptation, such as iNaturalist2018, recursive WA yields clear benefits by leveraging information from more data-rich subsets.

Following this intuition, in all of our experiments in the paper, we use only two values for λ: 0.3 and 0.7, corresponding to high and low adaptation needs, respectively. Intuitively, when the target dataset is close to the pre-training weights, the value of the λ becomes less important as even small datasets are enough for adaptations. However, when the shift becomes larger, subsets with more data (albeit biased towards head classes) become crucial.

Effect of class-balance strategies. By default, LT-Soups exhibits partial insensitivity to the choice of loss function, due to the balance introduced through subsampling and model averaging. However, in the first stage, each subset remains mildly imbalanced, though significantly less so than the full training set. Table 6 reports the impact of different class-balancing strategies used during this

Table 6: Comparison of PEFT and LT-Soups under different loss functions (CE, CB, LA).

Methods	CE			CB			LA		
	All	Head	Tail	All	Head	Tail	All	Head	Tail
PEFT	72.6	85.1	65.9	75.3	81.0	72.2	77.1	83.0	73.9
LT-Soups	76.3	84.5	71.9	78.2	84.5	78.4	78.6	85.0	75.2

stage, including logit adjustment loss (LA), cross-entropy loss (CE), and class-balanced sampling (CB) [25]. Unlike PEFT, which heavily depends on LA loss for optimal performance, LT-Soups is only moderately affected by the choice of class-balancing strategies, owing to the structural regularization introduced by training on multiple, complementary subsets.

Computational analysis. LT-Soups involves a total of $NM + 1$ training runs: M models are trained at each of the N subsampling levels in the first stage, followed by a single classifier trained on the full dataset in the second stage. Two key factors mitigate the computational burden of this procedure. First, each Stage 1 model is trained on a heavily subsampled dataset, significantly smaller than the full training set. For instance, under a maximum imbalance ratio of 64, each model sees only about 65% of the full dataset (see Table 11 in the Appendix for precise values), which leads to substantially faster training times compared to full-data training. Second, all models in the first stage are trained independently, enabling full parallelization. As a result, the overall wall-clock time is bounded by the longest individual training job, typically the one using the most imbalanced subset (e.g., imbalance ratio 64). This parallel-friendly design allows LT-Soups to scale efficiently and deliver competitive performance with minimal overhead. Appendix B provides a full breakdown of computational cost.

6 Limitations and Future Work

While we define the head-tail ratio using a fixed sample threshold, this binary split may oversimplify the class distribution structure. A more nuanced approach could leverage the generalized Pareto distribution [47] to model imbalance with controllable location, scale, and shape parameters. We leave exploration of such parameterized formulations for future work.

7 Conclusion

This work introduces LT-Soups, a novel two-stage model merging framework tailored for long-tailed distributions. We identify the head-tail ratio (η) as a critical yet underexplored factor influencing model performance alongside the commonly studied imbalance ratio (ρ). Through comprehensive experiments, we demonstrate that existing approaches, particularly PEFT and traditional model soups, fail to generalize across the full spectrum of imbalance scenarios. In contrast, LT-Soups builds a weight ensemble by averaging fully fine-tuned models trained on subsets with varying imbalance ratios, enabling specialization across the imbalance spectrum while preserving robust representations. Extensive experiments on five benchmarks and ablations on Tiny-ImageNet-LT confirm its consistent performance across long-tailed scenarios.

Acknowledgements. This work was supported by Distech Controls Inc., the Natural Sciences and Engineering Research Council of Canada, the Digital Research Alliance of Canada, and MITACS. This work was also supported in part by the project SERICS (PE00000014) under the NRRP MUR program funded by the EU - NGEU.

References

[1] Anton Alexandrov, Veselin Raychev, Mark Niklas Mueller, Ce Zhang, Martin Vechev, and Kristina Toutanova. Mitigating catastrophic forgetting in language transfer via model merging. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *EMNLP*, pages 17167–17186, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.1000. URL https://aclanthology.org/2024.findings-emnlp.1000/.

[2] Leo Breiman. Bagging predictors. *Machine learning*, 24:123–140, 1996.

[3] Glenn W Brier. Verification of forecasts expressed in terms of probability. *Monthly weather review*, 78(1):1–3, 1950.

[4] Jiarui Cai, Yizhou Wang, and Jenq-Neng Hwang. Ace: Ally complementary experts for solving long-tailed recognition in one-shot. In *ICCV*, pages 112–121, 2021.

[5] Kaidi Cao, Colin Wei, Adrien Gaidon, Nikos Arechiga, and Tengyu Ma. Learning imbalanced datasets with label-distribution-aware margin loss. *NeurIPS*, 32, 2019.

[6] Kamalika Chaudhuri, Kartik Ahuja, Martin Arjovsky, and David Lopez-Paz. Why does throwing away data improve worst-group error? In *ICML*, pages 4144–4188. PMLR, 2023.

[7] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16: 321–357, 2002.

[8] Shoufa Chen, Chongjian Ge, Zhan Tong, Jiangliu Wang, Yibing Song, Jue Wang, and Ping Luo. Adaptformer: Adapting vision transformers for scalable visual recognition. *NeurIPS*, 35: 16664–16678, 2022.

[9] Wuxing Chen, Kaixiang Yang, Zhiwen Yu, Yifan Shi, and CL Chen. A survey on imbalanced learning: latest research, applications and future directions. *Artificial Intelligence Review*, 57 (6):1–51, 2024.

[10] Jiequan Cui, Zhisheng Zhong, Shu Liu, Bei Yu, and Jiaya Jia. Parametric contrastive learning. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 715–724, 2021.

[11] Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. Class-balanced loss based on effective number of samples. In *CVPR*, pages 9268–9277, 2019.

[12] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255. Ieee, 2009.

[13] Bowen Dong, Pan Zhou, Shuicheng Yan, and Wangmeng Zuo. Lpt: Long-tailed prompt tuning for image classification. In *ICLR*, 2022.

[14] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021. URL `https://openreview.net/forum?id=YicbFdNTTy`.

[15] Arthur Douillard, Qixuan Feng, Andrei A Rusu, Rachita Chhaparia, Yani Donchev, Adhiguna Kuncoro, Marc'Aurelio Ranzato, Arthur Szlam, and Jiajun Shen. Diloco: Distributed low-communication training of language models. *arXiv preprint arXiv:2311.08105*, 2023.

[16] Jintong Gao, He Zhao, Zhuo Li, and Dandan Guo. Enhancing minority classes by mixing: An adaptative optimal transport approach for long-tailed classification. *Advances in neural information processing systems*, 36:60329–60348, 2023.

[17] Haibo He and Edwardo A Garcia. Learning from imbalanced data. *IEEE Transactions on knowledge and data engineering*, 21(9):1263–1284, 2009.

[18] Yin-Yin He, Jianxin Wu, and Xiu-Shen Wei. Distilling virtual examples for long-tailed recognition. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 235–244, 2021.

[19] Gregory Holste, Song Wang, Ziyu Jiang, Thomas C. Shen, George L. Shih, Ronald M. Summers, Yifan Peng, and Zhangyang Wang. Long-tailed classification of thorax diseases on chest x-ray: A new benchmark study. *MICCAI workshop, DALI (Workshop)*, 13567:22–32, 2022. URL `https://api.semanticscholar.org/CorpusID:251903892`.

[20] Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *ICLR*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

[21] Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E. Hopcroft, and Kilian Q. Weinberger. Snapshot ensembles: Train 1, get m for free. In *ICLR*, 2017. URL https://openreview.net/forum?id=BJYwwY9ll.

[22] Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. In *ICLR*, 2023. URL https://openreview.net/forum?id=6t0Kwf8-jrj.

[23] Pavel Izmailov, Dmitrii Podoprikhin, T. Garipov, Dmitry P. Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. In *Conference on Uncertainty in Artificial Intelligence*, 2018. URL https://api.semanticscholar.org/CorpusID:3833416.

[24] Pavel Izmailov, Polina Kirichenko, Nate Gruver, and Andrew G Wilson. On feature learning in the presence of spurious correlations. *NeurIPS*, 35:38516–38532, 2022.

[25] Bingyi Kang, Saining Xie, Marcus Rohrbach, Zhicheng Yan, Albert Gordo, Jiashi Feng, and Yannis Kalantidis. Decoupling representation and classifier for long-tailed recognition. In *ICLR*, 2020. URL https://openreview.net/forum?id=r1gRTCVFvB.

[26] Jaehyung Kim, Jongheon Jeong, and Jinwoo Shin. M2m: Imbalanced classification via major-to-minor translation. In *CVPR*, pages 13896–13905, 2020.

[27] Wei-Yin Ko, Daniel D'souza, Karina Nguyen, Randall Balestriero, and Sara Hooker. Fair-ensemble: When fairness naturally emerges from deep ensembling. *arXiv preprint arXiv:2303.00586*, 2023.

[28] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[29] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *NeurIPS*, 30, 2017.

[30] Ya Le and Xuan S. Yang. Tiny imagenet visual recognition challenge. 2015. URL https://api.semanticscholar.org/CorpusID:16664790.

[31] Jun Li, Zichang Tan, Jun Wan, Zhen Lei, and Guodong Guo. Nested collaborative learning for long-tailed visual recognition. In *CVPR*, pages 6949–6958, 2022.

[32] Weishi Li, Yong Peng, Miao Zhang, Liang Ding, Han Hu, and Li Shen. Deep model fusion: A survey. *arXiv preprint arXiv:2309.15698*, 2023.

[33] Xu-Ying Liu, Jianxin Wu, and Zhi-Hua Zhou. Exploratory undersampling for class-imbalance learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(2): 539–550, 2008.

[34] Ziwei Liu, Zhongqi Miao, Xiaohang Zhan, Jiayun Wang, Boqing Gong, and Stella X Yu. Large-scale long-tailed recognition in an open world. In *CVPR*, pages 2537–2546, 2019.

[35] Alexander Long, Wei Yin, Thalaiyasingam Ajanthan, Vu Nguyen, Pulak Purkait, Ravi Garg, Alan Blair, Chunhua Shen, and Anton van den Hengel. Retrieval augmented classification for long-tail visual recognition. In *CVPR*, pages 6959–6969, 2022.

[36] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

[37] Teli Ma, Shijie Geng, Mengmeng Wang, Jing Shao, Jiasen Lu, Hongsheng Li, Peng Gao, and Yu Qiao. A simple long-tailed recognition baseline via vision-language model. *arXiv preprint arXiv:2111.14745*, 2021.

[38] Yanbiao Ma, Licheng Jiao, Fang Liu, Shuyuan Yang, Xu Liu, and Lingling Li. Curvature-balanced feature manifold learning for long-tailed classification. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 15824–15835, 2023.

[39] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[40] Aditya Krishna Menon, Sadeep Jayasumana, Ankit Singh Rawat, Himanshu Jain, Andreas Veit, and Sanjiv Kumar. Long-tail learning via logit adjustment. In *ICLR*, 2021. URL `https://openreview.net/forum?id=37nvvqkCo5`.

[41] Jishnu Mukhoti, Yarin Gal, Philip HS Torr, and Puneet K Dokania. Fine-tuning can cripple your foundation model; preserving features may be the solution. *arXiv preprint arXiv:2308.13320*, 2023.

[42] Mahdi Pakdaman Naeini, Gregory Cooper, and Milos Hauskrecht. Obtaining well calibrated probabilities using bayesian binning. In *AAAI*, volume 29, 2015.

[43] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, pages 8748–8763. PMLR, 2021.

[44] Alexandre Rame, Matthieu Kirchmeyer, Thibaud Rahier, Alain Rakotomamonjy, Patrick Gallinari, and Matthieu Cord. Diverse weight averaging for out-of-distribution generalization. *NeurIPS*, 35:10821–10836, 2022.

[45] Alexandre Ramé, Johan Ferret, Nino Vieillard, Robert Dadashi, Léonard Hussenot, Pierre-Louis Cedoz, Pier Giuseppe Sessa, Sertan Girgin, Arthur Douillard, and Olivier Bachem. Warp: On the benefits of weight averaged rewarded policies. *arXiv preprint arXiv:2406.16768*, 2024.

[46] Jiawei Ren, Cunjun Yu, Xiao Ma, Haiyu Zhao, Shuai Yi, et al. Balanced meta-softmax for long-tailed visual recognition. *NeurIPS*, 33:4175–4186, 2020.

[47] Holger Rootzén and Nader Tajvidi. Multivariate generalized pareto distributions. *Bernoulli*, 12 (5):917–930, 2006.

[48] Jie Shao, Ke Zhu, Hanxiao Zhang, and Jianxin Wu. Diffult: Diffusion for long-tail recognition without external knowledge. *Advances in Neural Information Processing Systems*, 37:123007–123031, 2024.

[49] Jiang-Xin Shi, Tong Wei, Yuke Xiang, and Yu-Feng Li. How re-sampling helps for long-tail learning? *NeurIPS*, 36, 2023.

[50] Jiang-Xin Shi, Tong Wei, Zhi Zhou, Jie-Jing Shao, Xin-Yan Han, and Yu-Feng Li. Long-tail learning with foundation model: Heavy fine-tuning hurts. In *ICML*, 2024.

[51] Reece Shuttleworth, Jacob Andreas, Antonio Torralba, and Pratyusha Sharma. Lora vs full fine-tuning: An illusion of equivalence. *arXiv preprint arXiv:2410.21228*, 2024.

[52] Min-Kook Suh and Seung-Woo Seo. Long-tailed recognition by mutual information maximization between latent features and ground-truth labels. In *International conference on machine learning*, pages 32770–32782. PMLR, 2023.

[53] Yingfan Tao, Jingna Sun, Hao Yang, Li Chen, Xu Wang, Wenming Yang, Daniel Du, and Min Zheng. Local and global logit adjustments for long-tailed learning. In *ICCV*, pages 11783–11792, 2023.

[54] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. *NeurIPS*, 30, 2017.

[55] Changyao Tian, Wenhai Wang, Xizhou Zhu, Jifeng Dai, and Yu Qiao. Vl-ltr: Learning class-wise visual-linguistic representation for long-tailed visual recognition. In *ECCV*, pages 73–91. Springer, 2022.

[56] Grant Van Horn, Oisin Mac Aodha, Yang Song, Yin Cui, Chen Sun, Alex Shepard, Hartwig Adam, Pietro Perona, and Serge Belongie. The inaturalist species classification and detection dataset. In *CVPR*, pages 8769–8778, 2018.

[57] Xudong Wang, Long Lian, Zhongqi Miao, Ziwei Liu, and Stella Yu. Long-tailed recognition by routing diverse distribution-aware experts. In *ICLR*, 2021. URL https://openreview.net/forum?id=D9I3drBz4UC.

[58] Yidong Wang, Zhuohao Yu, Jindong Wang, Qiang Heng, Hao Chen, Wei Ye, Rui Xie, Xing Xie, and Shikun Zhang. Exploring vision-language models for imbalanced learning. *IJCV*, 132(1):224–237, August 2023. ISSN 0920-5691. doi: 10.1007/s11263-023-01868-w. URL https://doi.org/10.1007/s11263-023-01868-w.

[59] Xin Wen, Bingchen Zhao, Yilun Chen, Jiangmiao Pang, and XIAOJUAN QI. What makes CLIP more robust to long-tailed pre-training data? a controlled study for transferable insights. In *NeurIPS*, 2024. URL https://openreview.net/forum?id=PcyioHOmjq.

[60] Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *ICML*, pages 23965–23998. PMLR, 2022.

[61] Mitchell Wortsman, Gabriel Ilharco, Jong Wook Kim, Mike Li, Simon Kornblith, Rebecca Roelofs, Raphael Gontijo Lopes, Hannaneh Hajishirzi, Ali Farhadi, Hongseok Namkoong, et al. Robust fine-tuning of zero-shot models. In *CVPR*, pages 7959–7971, 2022.

[62] Liuyu Xiang, Guiguang Ding, and Jungong Han. Learning from multiple experts: Self-paced knowledge distillation for long-tailed classification. In *ECCV*, pages 247–263. Springer, 2020.

[63] Zhengzhuo Xu, Ruikang Liu, Shuo Yang, Zenghao Chai, and Chun Yuan. Learning imbalanced data with vision transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 15793–15803, 2023.

[64] Yuzhe Yang, Haoran Zhang, Dina Katabi, and Marzyeh Ghassemi. Change is hard: A closer look at subpopulation shift. In *ICML*, 2023.

[65] Zhiyong Yang, Qianqian Xu, Zitai Wang, Sicong Li, Boyu Han, Shilong Bao, Xiaochun Cao, and Qingming Huang. Harnessing hierarchical label distribution variations in test agnostic long-tail recognition. In *ICML*, 2024. URL https://openreview.net/forum?id=ebt5BfRHcW.

[66] Runtian Zhai, Chen Dan, J Zico Kolter, and Pradeep Kumar Ravikumar. Understanding why generalized reweighting does not improve over ERM. In *ICLR*, 2023. URL https://openreview.net/forum?id=ashPce_W8F-.

[67] Songyang Zhang, Zeming Li, Shipeng Yan, Xuming He, and Jian Sun. Distribution alignment: A unified framework for long-tail visual recognition. In *CVPR*, pages 2361–2370, 2021.

[68] Yifan Zhang, Bryan Hooi, Lanqing Hong, and Jiashi Feng. Self-supervised aggregation of diverse experts for test-agnostic long-tailed recognition. *Advances in neural information processing systems*, 35:34077–34090, 2022.

[69] Qihao Zhao, Chen Jiang, Wei Hu, Fan Zhang, and Jun Liu. Mdcs: More diverse experts with consistency self-distillation for long-tailed recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11597–11608, 2023.

[70] Qihao Zhao, Yalun Dai, Shen Lin, Wei Hu, Fan Zhang, and Jun Liu. Ltrl: Boosting long-tail recognition via reflective learning. In *European Conference on Computer Vision*, pages 1–18. Springer, 2024.

[71] Yan Zhao, Weicong Chen, Xu Tan, Kai Huang, and Jihong Zhu. Adaptive logit adjustment loss for long-tailed visual recognition. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pages 3472–3480, 2022.

[72] Zhisheng Zhong, Jiequan Cui, Shu Liu, and Jiaya Jia. Improving calibration for long-tailed recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16489–16498, 2021.

[73] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *PAMI*, 40(6):1452–1464, 2018. doi: 10.1109/TPAMI.2017.2723009.

[74] Boyan Zhou, Quan Cui, Xiu-Shen Wei, and Zhao-Min Chen. Bbn: Bilateral-branch network with cumulative learning for long-tailed visual recognition. In *CVPR*, pages 9719–9728, 2020.

[75] Jianggang Zhu, Zheng Wang, Jingjing Chen, Yi-Ping Phoebe Chen, and Yu-Gang Jiang. Balanced contrastive learning for long-tailed visual recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6908–6917, 2022.

Broader Impact

Our work advances the field of long-tailed recognition by improving model performance across imbalanced datasets, which are prevalent in real-world applications such as medical imaging, wildlife monitoring, and autonomous driving. By enhancing accuracy for both head and tail classes, our method promotes fairness and inclusivity in AI systems, reducing biases toward dominant categories.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Empirical validation on the synthetic CIFAR100 benchmark is partially presented in the Introduction and Section 3. The remaining experimental results are detailed in Section 5.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 6

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: No theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Section 5 and Appendix D

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: All of the datasets are publicly available. And the implementation details are provide in Appendix D

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: Appendix D and Appendix A

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: Due to resource constraints, all experiments were conducted using a single random seed. For model soups, however, multiple seeds were used during training, as required by the method, but only the averaged results are reported.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Appendix B

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Our work is based on public data.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Section 7

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: We only use public datasets.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [NA]

    Justification: The paper does not use existing assets.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: We used LLM only for writing, editing, or formatting purposes.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

A Additional ablations

Model calibration analysis. An inherent advantage of model merging methods is their ability to improve prediction calibration metrics. We evaluate LT-Soups against PEFT and Full-FT by measuring Negative Log-Likelihood (NLL), Expected Calibration Error (ECE) [42], and Brier score [3]. For NLL and Brier scores, we also provide category-wise results. All metrics are computed after temperature tuning on the validation set. As shown in 7, LT-Soups consistently outperforms the other methods on TinyImageNet-LT in terms of calibration.

Table 7: Calibration metrics on TinyImageNet for Full-FT, PEFT, and our LT-Soup.

Method	Metric	Overall	Head	Tail
Full-FT	ECE	1.97	-	-
	Brier Score	0.36	0.21	0.40
	NLL	1.03	0.63	1.25
PEFT	ECE	1.95	-	-
	Brier Score	0.32	0.23	0.35
	NLL	0.89	0.68	0.99
LT-Soups	ECE	**1.36**	-	-
	Brier Score	**0.30**	**0.20**	**0.33**
	NLL	**0.83**	**0.59**	**0.97**

Table 8: Performance across different values of λ with a fixed M=2, on TinyImageNet-LT.

	λ=0.3	λ=0.7
Acc	78.3	78.6
Head	84.6	85.0
Tail	75.0	75.2

Table 9: Performance across different values of M with a fixed λ=0.7, on TinyImageNet-LT.

	M=1	M=2	M=12
Acc	78.2	78.6	78.8
Head	84.8	85.0	85.5
Tail	74.6	75.2	75.2

PEFT compatibility. A natural question is whether PEFT methods can replace the full fine-tuning process in LT-Soups. To investigate this, we use LoRA as a representative approach. In the first stage, we freeze the CLIP pre-trained weights and tune LoRA parameters using the same subsets as LT-Soups. The LoRA parameters are combined with the pre-trained weights before applying our merging schema. Finally, we retrain the classifier using the LA loss. The performance on TinyImagenet-LT dataset (77.1 vs 77.2) matches that of end-to-end LoRA training. We hypothesize this outcome is due to a phenomenon observed in LLM literature [51], where LoRA introduces high-ranking singular vectors (intruder dimensions) that are absent in full fine-tuning. While these models achieve comparable task performance, they adapt less robustly to sequential tasks and diverge from the pre-training distribution.

Table 10: Effect of subsampling and classifier re-training in conjunction with the PEFT method. Each column reports PEFT fine-tuning performance on a given subsample ratio ρ after classifier re-training on TinyImageNet-LT.

ρ	1	2	4	8	16	32	64	100
All	73.9	74.3	74.8	76.2	76.4	77.1	77.0	77.0
Head	75.9	75.8	77.0	78.2	80.4	81.8	81.9	83.0
Tail	72.8	73.5	73.6	75.1	74.2	74.5	74.4	73.8

Hyperparameter sensitivity. In addition to the number of subsets used during LT-Soups fine-tuning, two other hyperparameters impact performance: (1) M: the number of models trained per subset D_{ρ_i}, with each model bootstrapped from the same imbalance ratio ρ_s. (2) λ: the interpolation coefficient used during recursive weight averaging.

Table 9 shows that increasing M on TinyImageNet-LT improves overall, head-, and tail-class accuracy, highlighting the benefits of ensembling across bootstrapped models. To ensure computational feasibility across all five datasets, we fix $M = 2$ in the main experiments.

Table 8 compares performance for $\lambda = 0.3$ and $\lambda = 0.7$. We observe that datasets closely aligned with CLIP's pretraining domain benefit from a larger λ, which retains more pretrained knowledge. In contrast, datasets with significant domain shifts—such as NIH-CXR-LT—perform better with smaller λ, allowing greater adaptation during model merging.

Table 11: Number of subsampling rounds (N), size of the largest subset relative to the full training set and λ used for each dataset.

Dataset	N	Relative size of largest subset	λ
CIFAR100-LT	5	67	0.7
Places-LT	5	63	0.7
ImageNet-LT	7	79	0.7
iNaturalist	8	90	0.3
NIH-CXR-LT	8	24	0.3

B Full Computational analysis

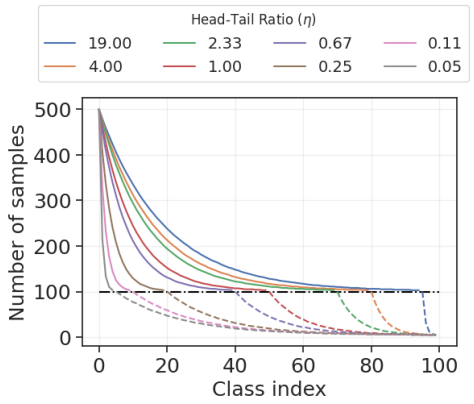
Table 12: Comparison of methods across ImageNet-LT and CXR-LT in terms of training time, iterations, model size, memory, and accuracy.

Method	Wall-clock Time (H-M-S)	Training Iterations	Parameters (M)	Memory (G)
ImageNet-LT				
Full-FT	1:37:56	9060	87.0	14.5
Model Soups	1:37:56	9060	87.0	14.5
LoRA ($rank = 64$)	1:25:33	9060	9.0	13.3
LT-Soups Stage 1	1:15:38	8050	87.0	14.5
LT-Soups Stage 2	0:30:00	9060	0.7	2.6
Full LT-Soups	1:48:38	–	–	–
NIH-CXR-LT				
Full-FT	0:53:43	5320	87.0	14.5
Model Soups	0:53:43	5320	87.0	14.5
LoRA ($rank = 64$)	2:14:32	13300	9.0	13.3
LT-Soups Stage 1	0:12:51	1300	87.0	14.5
LT-Soups Stage 2	0:19:26	5320	0.7	2.6
Full LT-Soups	0:32:17	–	–	–

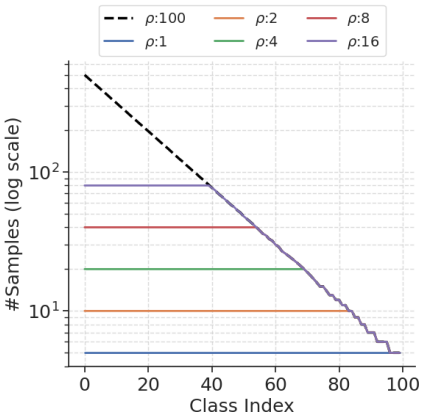
Table 12 compares the computational costs of Full Fine-Tuning (Full-FT), LIFT (which employs a LoRA adapter with rank 64 applied to all MLP layers), Model Soups, and LT-Soups on the ImageNet-LT and NIH-CXR-LT datasets. All models were trained to convergence using a batch size of 128 and mixed-precision training with NVIDIA RTX 3090 GPUs (24GB VRAM), using Python 3.9.15, PyTorch 2.4.0, and CUDA 11.8. For LT-Soups, we break down the computational cost into two stages. Stage 1 involves training models independently and in parallel on subsets with different imbalance ratios. Since the subset with the highest imbalance ratio contains the most training samples, it dominates the overall wall-clock time. Stage 2 retrains only the linear classifier on the full dataset—a highly efficient step, as it updates just a single linear layer. In our experiments, we used the same number of epochs for both stages of LT-Soups.

The computational overhead of LT-Soups compared to existing methods depends heavily on dataset characteristics, particularly the original imbalance ratio. For example, in ImageNet-LT, which has an imbalance ratio of 256, the largest subset used in Stage 1 accounts for 89% of the full training data, resulting in relatively higher wall-clock time. In contrast, on NIH-CXR-LT, with a much more extreme imbalance ratio of 6401, the largest Stage 1 subset represents only 24% of the dataset, leading to a 4.4× reduction in training time compared to Full-FT (Table 11). Additionally, while full-rank

methods like Full-FT and LT-Soups typically converge within 10 epochs on CXR-LT, LIFT required 50 epochs—substantially increasing its wall-clock time despite its parameter-efficient design.



(a) Visualization of imbalance distributions in CIFAR100-LT with varying values of η.



(b) Example of subsampled distributions used in LT-Soups, with the x-axis shown on a logarithmic scale.

C Extended Related Work

C.1 Imbalanced Classification

We can roughly divide progress on imbalanced classification into three groups.

Re-sampling/Re-weighting. Class imbalance mitigation strategies broadly involve oversampling minority classes [7], subsampling majority classes [33], or reweighting losses [17]. Generative approaches such as DiffuLT [48] train a diffusion model on a long-tailed distribution and then uses it to generate a balanced proxy dataset for training the final classifier. Subsampling risks losing majority-class discriminative patterns, oversampling may overfit minority classes [74], and reweighting struggles in overparameterized networks [66]. Recent advances like *balanced softmax* [46] and its generalization, *logit adjustment* loss (LA) [40] address these issues by enforcing larger margins for tail classes, bridging data imbalance with geometric regularization.

Decoupled learning. Decoupled learning frameworks address class imbalance through sequential training phases: representation learning via instance-balanced sampling followed by classifier refinement using class-balanced strategies [25, 67]. This paradigm assumes model biases primarily reside in the classifier layer, positing that head-tail performance gaps can be resolved through post-hoc classifier calibration [24, 64]. However, [50] show empirically that this assumption becomes invalid when fine-tuning foundation models, neglecting a tailored strategy for representation learning, degrades both head and tail class performance due to catastrophic forgetting of pre-trained features [41].

Ensemble learning. Ensemble methods address data imbalance by combining specialized experts trained on complementary distributions [4, 31]. Notable approaches include: BBN's dual-branch architecture balancing original and re-sampled distributions [74]; RIDE's dynamic routing of instances to distribution-aware experts [57]; and LFME's multi-teacher distillation across many/medium/few-shot groups [62]; Reflective Learning [70] promotes consistency across training iterations by minimizing KL divergence between predictions and soft labels induced from feature similarity. While effective, these methods rely on heuristic expert specialization rules and often result in cumbersome architectures that hinder adaptation to foundation models, increase training complexity, and limit inference speed. Our work circumvents these limitations through two key innovations: (1) replacing specialized expert design with parallel fine-tuning of foundation models on controlled subsamples, and (2) employing model averaging and EMA instead of complex aggregation mechanisms. This pre-

serves the ensemble's variance-reduction benefits while maintaining the original foundation model's architectural simplicity and computational efficiency.

C.2 Model Merging

Model merging, also sometimes referred to to weight averaging, has gained significant attention in recent years as a promising research direction [32], focusing on reducing communication costs in federated learning [39] and distributed training [15], enabling the efficient combination of multiple models without additional training [22], and enhancing model robustness in out-of-distribution scenarios [60, 44]. Early approaches like Exponential Moving Average (EMA) [54] and Stochastic Weight Averaging (SWA) [23] have been widely adopted to accelerate training convergence, stability, and enhance the generalization capabilities of deep neural networks. Recent work extends merging to sequential adaptation: Alexandrov et al. [1] mitigates catastrophic forgetting in continual pretraining via iterative merging, while Ramé et al. [45] align LLMs through multi-stage averaging during RLHF. To our knowledge, no prior work applies model merging to imbalanced recognition. Unlike existing sequential merging approaches, our framework trains multiple models in parallel on complementary subsampled distributions, a critical design choice for handling long-tailed data. We propose the first schema specifically tailored for imbalance, integrating subsampling (to retain tail-class discriminability) and bootstrapping (to stabilize head-class representations). This parallelized merging strategy directly addresses feature-space asymmetry in long-tailed distributions while maintaining computational efficiency, enabling foundation models to adapt to extreme imbalance without sacrificing pre-trained generalization.

D Baselines and implementation details.

We use CLIP with the ViT-B/16 backbone. Following [43], we adopt a prototypical classification head for g, where both features and classifier weights are l_2-normalized, and a temperature is applied to the logits. The parameters ω are initialized by generating text. We use descriptive prompts such as "a photo of a cat" or "a photo of a dog" to represent each class [43]. for the classes and extracting corresponding textual features using the CLIP text encoder.

We optimize the model using the AdamW optimizer [36]. The batch size is set to 128, with learning rates of $3e-4$ for both the representation and the classification stage. A cosine decay learning rate scheduler is employed, gradually reducing the learning rate to $0.1 \cdot max\_lr$ after a warmup period spanning $max(100, 0.01 \cdot total\_steps)$ steps. The validation set of each dataset is used to select the best checkpoint. Table 11 shows the hyperparameters we used for each dataset. We select N and λ based on the validation set of each dataset and fix M=2 across all experiments. We report all baseline results without test-time augmentation, which offers orthogonal gains.

Table 13: Dataset details used in our work.

Dataset	Classes	Total samples	Max samples	Min samples	ρ	η
CIFAR100-LT [5]	100	10.8k	500	5	100	0.54
TinyImageNet-LT [30]	200	21.5k	500	5	100	0.53
Places-LT [34]	365	62.5k	4980	5	996	0.55
ImageNet-LT [34]	1000	115.8k	1280	5	256	0.62
iNaturalist [56]	8,142	437.5k	1000	2	500	0.11
NIH-CXR-LT [19]	20	88,637	53260	12	6491	5.66

D.1 Full results

Table 14: Comparison of methods for training on CIFAR100-LT.

Methods	Backbone	Overall	Many	Medium	Few
Training from scratch					
LDAM [5]	ResNet-32	42.0	-	-	-
BBN [74]	ResNet-32	42.6	-	-	-
DiVE [18]	ResNet-32	45.4	-	-	-
MiSLAS [72]	ResNet-32	47.0	-	-	-
BS [46]	ResNet-32	50.8	-	-	-
PaCo [10]	ResNet-32	52.0	-	-	-
BCL [75]	ResNet-32	51.9	-	-	-
Fine-tuning CLIP					
Linear Prob (LA)	ViT-B/16	70.0	77.2	71.1	60.4
Full-FT (LA)	ViT-B/16	79.6	88.1	79.9	69.3
cRT [25]	ViT-B/16	78.8	89.7	79.7	65.1
PEFT [50]	ViT-B/16	81.3	85.2	80.9	77.1
Model Soups [60]	ViT-B/16	82.1	**89.9**	82.2	73.0
LT-Soups (Ours)	ViT-B/16	**83.5**	88.2	**83.5**	**78.0**

Table 15: Comparison of methods for training on Places-LT.

Methods	Backbone	Overall	Many	Medium	Few
Training from ImageNet-1K pre-trained backbone					
OLTR [34]	ResNet-152	35.9	44.7	37.0	25.3
cRT [25]	ResNet-152	36.7	42.0	37.6	26.4
LWS [25]	ResNet-152	37.6	40.6	39.1	28.6
MiSLAS [72]	ResNet-152	40.4	39.6	43.3	36.1
DisAlign [67]	ResNet-152	39.3	40.4	39.4	32.9
ALA [71]	ResNet-152	41.2	36.1	47.9	35.3
PaCo [10]	ResNet-152	40.5	33.7	44.4	35.3
LiVT [63]	ViT-B/16	40.8	48.1	40.6	27.5
Fine-tuning CLIP					
Linear Prob (LA)	ViT-B/16	48.8	48.8	49.7	47.1
cRT [25]	ViT-B/16	44.4	51.0	43.1	35.4
BALLAD [37]	ViT-B/16	49.5	49.3	50.2	48.4
Decoder [58]	ViT-B/16	46.8	-	-	-
LPT [13]	ViT-B/16	50.1	49.3	52.3	46.9
Full-FT (LA)	ViT-B/16	46.6	49.9	46.3	41.4
cRT [25]	ViT-B/16	44.4	51.0	43.1	35.4
LIFT [50]	ViT-B/16	**51.5**	51.3	52.2	**50.5**
Model Soups [60]	ViT-B/16	49.4	**51.7**	50.0	43.7
LT-Soups (Ours)	ViT-B/16	**51.7**	51.2	**52.8**	50.3

Table 16: Comparison of methods for training on ImageNet-LT.

Methods	Backbone	Overall	Many	Medium	Few
Training from scratch					
cRT [25]	ResNet-50	47.3	58.8	44.0	26.1
LWS [25]	ResNet-50	47.7	57.1	45.2	29.3
MiSLAS [72]	ResNet-50	52.7	62.9	50.7	31.0
LA [40]	ResNet-50	51.1	-	-	-
DisAlign [67]	ResNet-50	52.9	61.3	52.2	31.4
BCL [75]	ResNet-50	56.0	-	-	-
PaCo [10]	ResNet-50	57.0	-	-	-
NCL [31]	ResNet-50	57.4	-	-	-
LiVT [63]	ViT-B/16	60.9	73.6	56.4	41.0
Fine-tuning CLIP					
Linear Prob (LA)	ViT-B/16	74.2	77.8	73.3	67.4
BALLAD [37]	ViT-B/16	75.7	79.1	74.5	69.8
Decoder [58]	ViT-B/16	73.2	-	-	-
Full-FT (LA)	ViT-B/16	73.9	79.8	71.9	63.9
cRT [25]	ViT-B/16	72.6	81.1	70.6	56.1
LIFT [50]	ViT-B/16	<u>77.0</u>	80.2	**76.1**	**71.5**
Model Soups [60]	ViT-B/16	76.0	**81.5**	<u>74.5</u>	65.5
LT-Soups (Ours)	ViT-B/16	**77.4**	<u>81.2</u>	**76.1**	<u>70.7</u>

Table 17: Comparison of methods for training on NIH-CXR-LT.

Methods	Backbone	Overall	Many	Medium	Few
Training from ImageNet-1K pre-trained backbone					
cRT [25]	ResNet-50	38.0	43.3	37.4	30.0
LWS [25]	ResNet-50	28.0	<u>45.7</u>	23.0	08.3
CB LDAM-DRW [5]	ResNet-50	37.7	**47.6**	35.6	25.0
CB Softmax [11]	ResNet-50	33.3	29.5	**41.5**	21.7
Fine-tuning CLIP					
Linear Prob (LA)	ViT-B/16	17.5	13.3	21.1	16.7
BALLAD [37]	ViT-B/16	34.5	36.7	38.9	20.8
Full-FT (LA)	ViT-B/16	38.0	43.8	**41.5**	20.0
cRT [25]	ViT-B/16	37.7	42.9	39.3	25.0
LIFT [50]	ViT-B/16	<u>38.5</u>	43.3	40.4	<u>25.5</u>
Model Soups [60]	ViT-B/16	38.0	45.6	40.2	20.0
LT-Soups (Ours)	ViT-B/16	**39.3**	42.4	<u>40.7</u>	**30.8**

Table 18: Comparison of methods for training on iNaturalist 2018.

Methods	Backbone	Overall	Many	Medium	Few
Training from scratch					
cRT [25]	ResNet-50	65.2	69.0	66.0	63.2
LWS [25]	ResNet-50	65.9	65.0	66.3	65.5
MiSLAS [72]	ResNet-50	71.6	73.2	72.4	70.4
DiVE [18]	ResNet-50	69.1	70.6	70.0	67.7
DisAlign [67]	ResNet-50	69.5	69.1	69.9	69.4
ALA [71]	ResNet-50	69.6	69.5	70.2	69.0
RIDE [57]	ResNet-50	71.5	72.4	73.1	70.4
RIDE+CR [38]	ResNet-50	73.5	74.0	74.3	73.1
RIDE+OTmix [16]	ResNet-50	73.7	74.1	75.2	72.8
BCL [75]	ResNet-50	71.8	-	-	-
PaCo [10]	ResNet-50	73.2	70.4	72.8	75.8
NCL [31]	ResNet-50	74.2	72.0	74.9	73.8
GML [52]	ResNet-50	74.5	-	-	-
LiVT [63]	ViT-B/16	76.1	**78.9**	76.5	74.8
Fine-tuning CLIP					
Linear Prob (LA)	ViT-B/16	60.4	48.9	60.0	63.9
Decoder [58]	ViT-B/16	59.2	-	-	-
LPT [13]	ViT-B/16	76.1	-	-	79.3
Full-FT (LA)	ViT-B/16	76.1	75.7	76.9	75.3
LIFT [50]	ViT-B/16	**79.1**	72.4	**79.0**	**81.1**
Model Soups [60]	ViT-B/16	76.4	77.1	76.8	75.6
LT-Soups (Ours)	ViT-B/16	<u>78.2</u>	<u>76.7</u>	<u>78.5</u>	<u>78.2</u>