Long-Tail Learning with Foundation Model: Heavy Fine-Tuning Hurts

Jiang-Xin Shi [*1,2] **Tong Wei** [*3,4] **Zhi Zhou** [1] **Jie-Jing Shao** [1] **Xin-Yan Han** [1] **Yu-Feng Li** [1,2]

Abstract

The fine-tuning paradigm in addressing long-tail learning tasks has sparked significant interest since the emergence of foundation models. Nonetheless, how fine-tuning impacts performance in long-tail learning was not explicitly quantified. In this paper, we disclose that heavy fine-tuning may even lead to non-negligible performance deterioration on tail classes, and lightweight fine-tuning is more effective. The reason is attributed to inconsistent class conditions caused by heavy fine-tuning. With the observation above, we develop a low-complexity and accurate long-tail learning algorithms LIFT with the goal of facilitating fast prediction and compact models by adaptive lightweight fine-tuning. Experiments clearly verify that both the training time and the learned parameters are significantly reduced with more accurate predictive performance compared with state-of-the-art approaches. The implementation code is available at https://github.com/shijxcs/LIFT.

1. Introduction

Long-tail learning addresses the challenge of learning from highly imbalanced data, where a small set of classes (head classes) is well-represented in the training data, while other classes (tail classes) have only a limited number of training samples available. Given its widespread attention, numerous long-tail learning approaches have emerged to enhance generalization, particularly for tail classes. These approaches typically fall into three categories: 1) data manipulation (Zhou et al., 2020; Kang et al., 2020), 2) representation learning (Zhong et al., 2021; Cui et al., 2021), and 3) model output adjustment (Ren et al., 2020; Menon et al., 2021). While these methods have made substantial strides, a significant gap still persists compared to models trained on class-balanced datasets.

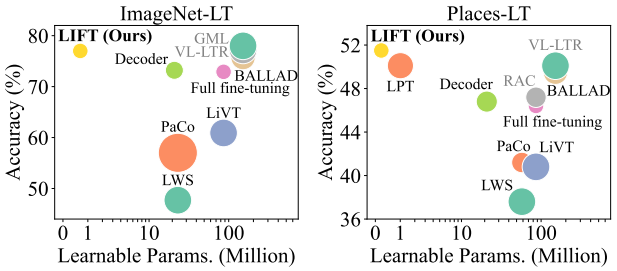


Figure 1: Comparison with state-of-the-art methods. The x-axis represents the number of learnable parameters and the y-axis shows the test accuracy. **The size of each point corresponds to the number of training epochs**, with larger points indicating longer training time. Gray labels denote methods that incorporate external data. LIFT consistently achieves higher performance with lower costs and is even comparable with methods that leverage external data.

Instead of training deep neural networks from scratch, recent results from BALLAD (Ma et al., 2021), RAC (Long et al., 2022), VL-LTR (Tian et al., 2022), and LPT (Dong et al., 2023) show that proper fine-tuning of foundation models such as CLIP (Radford et al., 2021) can surprisingly improve long-tail learning performance. For example, BALLAD first fully fine-tunes the foundation model, then freezes the backbone and optimizes a linear adapter on the re-sampled data. VL-LTR incorporates additional image-text web data during the fine-tuning process. RAC jointly fine-tunes an encoder and trains a retrieval module to augment the input image with external datasets such as ImageNet-21K. LPT fine-tunes the foundation model utilizing prompt tuning (Jia et al., 2022) via two-phrase training.

While these works introduce a new paradigm for long-tail learning, how fine-tuning affects performance in long-tail learning remains unexplored. In this paper, we reveal that heavy fine-tuning may lead to non-negligible performance deterioration on tail classes. To uncover the reasons behind it, we identify that full fine-tuning distorts the intra-

[*]Equal contribution [1]National Key Laboratory for Novel Software Technology, Nanjing University, China [2]School of Artificial Intelligence, Nanjing University, China [3]School of Computer Science and Engineering, Southeast University, China [4]Key Laboratory of Computer Network and Information Integration, Southeast University, Ministry of Education, China. Correspondence to: Yu-Feng Li <liyf@nju.edu.cn>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

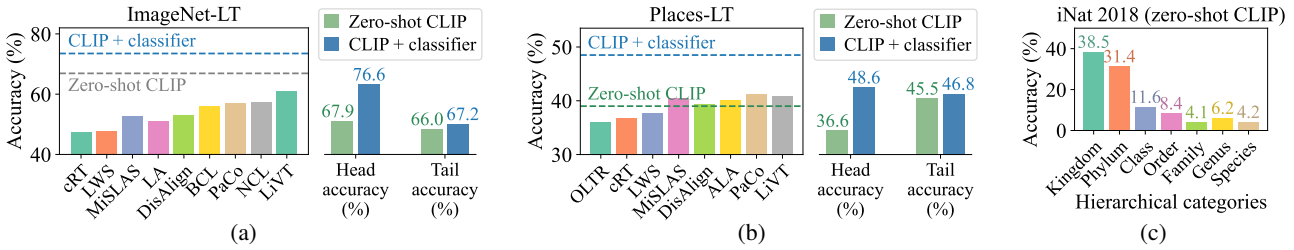


Figure 2: (a-b) On ImageNet-LT and Places-LT, zero-shot CLIP has surpassed many prior methods. By simply introducing an additional classifier, the accuracy further increases. However, the improvements mainly come from the head classes, while the tail classes only achieve marginal enhancements. (c) On iNaturalist 2018, zero-shot CLIP encounters challenges in achieving high accuracy for fine-grained long-tail categories.

class distance distributions. In theory, we demonstrate that such distortions break the consistency assumption of class-conditions, consequently resulting in biased predictions. Based on the above analysis and observation, we realize that optimizing a small proportion of the pre-trained weights may not only enhance discriminative capacity, but also leave the intra-class distributions to be unaffected. To this end, we propose a low-complexity and accurate long-tail learning method named *LIghtweight Fine-Tuning* (LIFT) with the goal of facilitating fast prediction and compact models by adaptive lightweight fine-tuning.

To achieve performance improvements and balanced prediction, previous work usually comes at the cost with 1) long training epochs (≈ 100); 2) a two-staged procedure; and 3) an external training dataset ($size \approx 10^6$). To overcome it, LIFT is a single-staged framework, which achieves convergence in fewer than 20 training epochs without requiring external training data. Furthermore, we introduce a semantic-aware classifier initialization to equip the model with a robust starting point. Moreover, we employ a test-time ensembling to compensate for the intrinsic defects of foundation models and enhance the generalization. LIFT brings small computational overhead, while consistently outperforming the state-of-the-art methods on a series of long-tail benchmark datasets. Figure 1 gives a performance comparison on two typical datasets ImageNet-LT and Places-LT.

The contributions of this paper are summarized as follows: **1)** We identify a limitation in heavy fine-tuning distorts the tail-class performance; **2)** We discover that optimizing a small proportion of parameters helps; **3)** We introduce a low-complexity and accurate long-tail learning method LIFT with lightweight fine-tuning; **4)** Comprehensive experiments demonstrate that LIFT consistently outperforms the state-of-the-art methods with a lower computational cost.

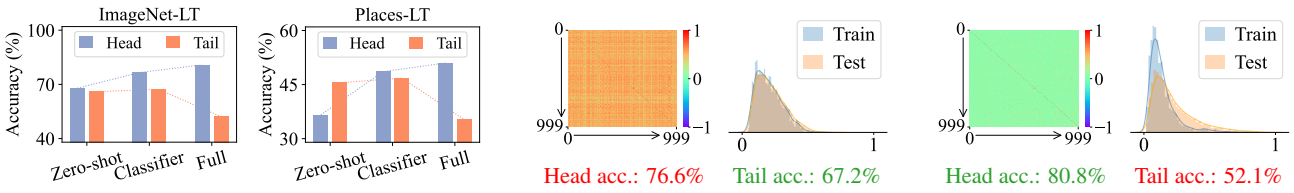
2. Long-Tail Learning with Foundation Model

Preliminary. Different from conventional neural networks, in the foundation models, the Transformer architecture (Vaswani et al., 2017; Dosovitskiy et al., 2021) is more simply designed while exhibiting remarkable generalization capabilities. It has been proven adaptable to a wide range of computer vision and natural language processing tasks (Devlin et al., 2019; Dosovitskiy et al., 2021). Notably, the recent vision-language pre-training model, CLIP, further underscores its efficacy by demonstrating impressive zero-shot performance (Radford et al., 2021). When processing an image $\boldsymbol{x}$ and considering K candidate classes, CLIP first generates the textual prompts for the classes. These prompts are descriptive phrases, such as "a photo of a cat" or "a photo of a dog". CLIP extracts corresponding textual features $\boldsymbol{f}_{T_1}, \cdots, \boldsymbol{f}_{T_K}$ and the image feature $\boldsymbol{f}_I$. To predict the label for the given image, CLIP compares the cosine similarity between the image and each of the class prompts:

$$y_{\text{pred}} = \arg\max_{k \in [K]} \frac{\boldsymbol{f}_I^\top \boldsymbol{f}_{T_k}}{\|\boldsymbol{f}_I\|_2 \|\boldsymbol{f}_{T_k}\|_2} \tag{1}$$

Long-Tail Learning with CLIP. In Figures 2a to 2c, we evaluate the performance of CLIP on typical long-tail datasets. We discover that zero-shot CLIP outperforms most conventional methods. Furthermore, by freezing the backbone and learning an additional classifier, the performance can be further improved, thereby highlighting the effectiveness of the representations learned by CLIP. Nonetheless, the majority of performance gains are dominated by the head classes, while the tail classes struggle to achieve comparable improvements.

Moreover, while zero-shot CLIP shows impressive performance, its good performance cannot be stably generalized to common long-tail datasets. For example, in iNaturalist 2018 dataset, it poses a fine-grained long-tail challenge, featuring a hierarchical categorization system spanning from 7 kingdoms to 8142 species. Although zero-shot CLIP achieves high accuracy for predicting coarse-grained categories like "kingdom" and "phylum", it performs poorly in predicting fine-grained species, *i.e.,* long-tail classes.



(a) Performance of fine-tuning. (b) Classifier fine-tuning. (c) Full fine-tuning.

Figure 3: (a) Full fine-tuning improves head-class accuracy while decreasing tail-class accuracy, even if we optimize the balanced LA loss. (b-c) Inter-class feature similarities (heatmaps) and intra-class distributions from tail classes (histograms) on ImageNet-LT. Classifier fine-tuning limits head-class performance due to unoptimized inter-class similarities. Full fine-tuning optimizes inter-class similarities but leads to inconsistent distribution between train and test data on tail classes.

3. Heavy Fine-Tuning Hurts

Although the foundation model has shown commendable performance in downstream long-tail learning tasks, it has indeed some limitations. As discussed earlier, CLIP has achieved high accuracy for head classes, but it has not yet achieved strong accuracy for tail classes. This naturally raises a question, is the fine-tuning not sufficient?

However, the results in Figure 3a show that full fine-tuning yields improvements in head-class accuracy, while at the expense of a reduction in tail-class accuracy. Note that we have already optimized the balanced logit-adjusted (LA) loss (Menon et al., 2021) and applied a balanced classifier initialization (will be introduced in Section 4.3), but the prediction results are still biased. In fact, performance deterioration on tail classes makes the overall accuracy of full fine-tuning even worse than that of classifier fine-tuning (72.9% vs. 73.5% on ImageNet-LT, and 46.4% vs. 48.5% on Places-LT).

To uncover the reasons behind it, we identify that full fine-tuning may distort the intra-class distance distributions. Specifically, we calculate the inter-class feature similarities and the intra-class distance distributions on ImageNet-LT and report the results in Figure 3. Notably, Figure 3c shows that full fine-tuning yields more discriminative representations as it reduces the inter-class similarities to approximately zero, which means the features of different classes are almost orthogonal. However, it also distorts the intra-class distribution, leading to a distribution shift between train and test data on tail classes. In this case, using the fine-tuned model to estimate the unknown tail-class data will inevitably lead to an underestimated class-conditional probability. We have conducted the following analysis on this matter.

Proposition 3.1. *Underestimated class-conditional probability* $\mathrm{P}(\boldsymbol{x} \mid y=j)$ *leads to an underestimated loss on class* j *and a biased prediction towards other classes.*

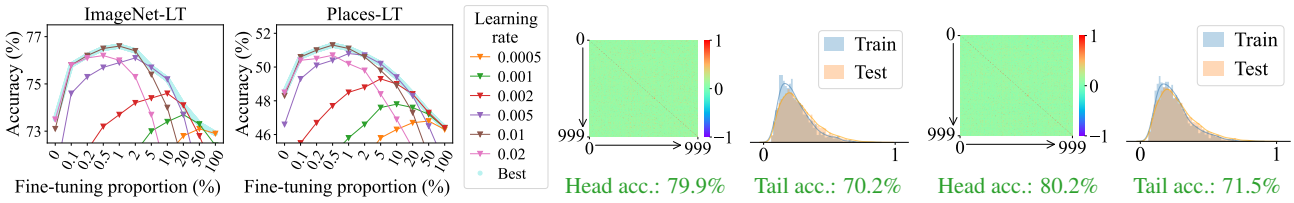
Proof. Denote P_s and P_t as the probability distribution in the source (training) and target (test) domain, respectively. For long-tail learning, $\mathrm{P}_s(y)$ appears a long-tail distribution and $\mathrm{P}_t(y)$ is a uniform distribution, *i.e.*, $\mathrm{P}_t(y=k) \equiv 1/K$. We then have,

$$\begin{aligned}\mathrm{P}_s(y=j \mid \boldsymbol{x}) &= \frac{\mathrm{P}_t(y=j \mid \boldsymbol{x}) \cdot \frac{\mathrm{P}_s(y=j \mid \boldsymbol{x})}{\mathrm{P}_t(y=j \mid \boldsymbol{x})}}{\sum_{k\in[K]} \mathrm{P}_t(y=k \mid \boldsymbol{x}) \cdot \frac{\mathrm{P}_s(y=k \mid \boldsymbol{x})}{\mathrm{P}_t(y=k \mid \boldsymbol{x})}} \\ &= \frac{\mathrm{P}_t(y=j \mid \boldsymbol{x}) \cdot \frac{\mathrm{P}_s(\boldsymbol{x} \mid y=j)}{\mathrm{P}_t(\boldsymbol{x} \mid y=j)} \cdot \frac{\mathrm{P}_s(y=j)}{\mathrm{P}_t(y=j)}}{\sum_{k\in[K]} \mathrm{P}_t(y=k \mid \boldsymbol{x}) \cdot \frac{\mathrm{P}_s(\boldsymbol{x} \mid y=k)}{\mathrm{P}_t(\boldsymbol{x} \mid y=k)} \cdot \frac{\mathrm{P}_s(y=k)}{\mathrm{P}_t(y=k)}} \\ &= \frac{\mathrm{P}_t(y=j \mid \boldsymbol{x}) \cdot \mathrm{P}_s(y=j) \cdot \zeta_{s-t}(j)}{\sum_{k\in[K]} \mathrm{P}_t(y=k \mid \boldsymbol{x}) \cdot \mathrm{P}_s(y=k) \cdot \zeta_{s-t}(k)} \end{aligned} \tag{2}$$

$$\begin{aligned}\mathcal{L}(\boldsymbol{x}, y=j) &= -\log \mathrm{P}_s(y=j \mid \boldsymbol{x}) \\ &= -\log \frac{\exp\left(z_j + \log \mathrm{P}_s(y=j) + \log \zeta_{s-t}(j)\right)}{\sum_{k\in[K]} \exp\left(z_k + \log \mathrm{P}_s(y=k) + \log \zeta_{s-t}(k)\right)} \end{aligned} \tag{3}$$

where $\zeta_{s-t}(j) = \dfrac{\mathrm{P}_s(\boldsymbol{x} \mid y=j)}{\mathrm{P}_t(\boldsymbol{x} \mid y=j)}$ and z_j is the predicted logit on class j. For samples $\boldsymbol{x}$ from class j, the underestimated $\mathrm{P}_t(\boldsymbol{x} \mid y=j)$ results in an underestimation of $\mathcal{L}(\boldsymbol{x}, y=j)$, and consequently leads to a biased optimization towards other classes. A more detailed proof is presented in Appendix B. □

Remark 3.2. Proposition 3.1 indicates that the performance deterioration of full fine-tuning is attributed to the inconsistent class-conditional distributions among the tail classes. Previous works such as LA (Menon et al., 2021) assume that the class-conditional distribution is consistent between the source and target domains, *i.e.*, $\zeta_{s-t}(j) = 1$. However, our empirical findings in Figure 3c reveal that full fine-tuning breaks this assumption. An ideal solution is to estimate the underlying class-conditional distribution; however, this is unfeasible due to the scarcity of tail-class data. Another approach is to prevent such distribution distortions. To achieve this goal, we introduce lightweight fine-tuning in Section 4.



(a) Fine-tuning with varying proportions. (b) Arbitrary lightweight fine-tuning. (c) Structured lightweight fine-tuning.

Figure 4: (a) Fine-tuning a small proportion of all parameters (*e.g.,* 0.1%-2%) yields superior performance. As the proportion increases, performance deteriorates even when we search for the best learning rate. (b-c) Inter-class feature similarities (heatmaps) and intra-class distributions from tail classes (histograms) on ImageNet-LT. Both arbitrary and structured lightweight fine-tuning perform well in optimizing inter-class similarities and preserving intra-class distributions.

In addition to the aforementioned issue, full fine-tuning also tends to encounter severe overfitting on long-tail datasets, particularly on the tail classes. We quantitatively assess the overfitting issue of full fine-tuning, and present the results in Appendix C due to the space constraint. To mitigate the issue of performance deterioration on tail classes, recent approaches have explored two-stage training procedures (Ma et al., 2021) or the inclusion of additional training data (Tian et al., 2022; Long et al., 2022). However, these strategies often introduce significant training overhead or require external data, thereby limiting their practicality. In response to this, we introduce LIFT, an efficient and accurate lightweight fine-tuning framework tailored for long-tail learning.

4. Efficient and Accurate Long-Tail Learning

4.1. Lightweight Fine-Tuning Helps

To alleviate the distortion of intra-class distributions, a direct method is to constrain the number of learnable parameters. Formally, for each weight matrix $\boldsymbol{W} \in \mathbb{R}^{d_1 \times d_2}$ in the foundation model, we optimizes a specified proportion α of parameters within $\boldsymbol{W}$, while keeping the rest frozen. Thus, only $\alpha d_1 d_2$ parameters are optimized. In practice, a sparse 0-1 mask $\boldsymbol{M} \in \{0,1\}^{d_1 \times d_2}$ is used to control the optimized parameters:

$$\boldsymbol{X}\boldsymbol{W} \rightarrow \boldsymbol{X}(\boldsymbol{W} \circ \boldsymbol{M}) + \underbrace{\boldsymbol{X}(\boldsymbol{W} \circ (1 - \boldsymbol{M}))}_{\text{gradient detached}} \quad (4)$$

where $\|\boldsymbol{M}\|_0 = \alpha d_1 d_2$. In Figure 4a, we investigate the impact of varying proportions of fine-tuned parameters. The result demonstrates the benefits of lightweight fine-tuning since only a small proportion (*e.g.,* 0.1%) yields a substantial improvement. As the proportion increases, performance faces a risk of degradation, highlighting the drawbacks of heavy fine-tuning. Besides, heavy fine-tuning is sensitive to hyperparameters, as it requires searching for an optimal learning rate.

It is noteworthy that the optimized parameters are selected arbitrarily, but the performance improvement is remarkable. This indicates that lightweight fine-tuning is crucial for enhancing long-tail learning, even without specific fine-tuning strategies. To get a deeper understanding, we visualize the inter-class feature similarities and the intra-class distributions in Figure 4b. The results demonstrate that arbitrary lightweight fine-tuning yields comparable feature separability with full fine-tuning, while its class conditions are preserved to be consistent between training and test data. Furthermore, prediction accuracy verifies the adaptation and generalization ability of arbitrary lightweight fine-tuning, as it achieves the same high performance as full-fine-tuning on head classes, and its tail-class performance is even superior.

4.2. The Proposed Fine-Tuning (LIFT) Method

We have demonstrated that arbitrary lightweight fine-tuning, even without any guidance for parameter selection, performs better than full fine-tuning. This indicates that a small learnable parameter quantity is more important for fine-tuning. To this end, we first investigate *structured lightweight fine-tuning* which learns a small set of task-specific parameters in a structured manner. Some related ideas, although already used in foundation models, are not directly suitable to long-tail learning. For more detailed introductions, please refer to Appendix D.

Figure 4c shows the potential of a very simple idea for structured lightweight fine-tuning. It optimizes the inter-class similarities to be nearly orthogonal and preserves the intra-class distributions undistorted. As can be seen, its performance already surpasses arbitrary lightweight fine-tuning on both head and tail classes. The findings indicate that more well-designed structured lightweight fine-tuning will further enhance the feature separability as well as ensure the consistency of class-conditional distributions, and will be more accurate.

Our analysis above justifies that lightweight fine-tuning can significantly mitigate performance deterioration and improve generalization. To this end, we propose a low-complexity and accurate long-tail learning method termed

LIghtweight Fine-Tuning (LIFT). LIFT is versatile and inclusive, allowing for the incorporation of a range of structured lightweight fine-tuning methods.

In order to have better adaptability among tasks, apart from the lightweight fine-tuning, a classifier is introduced to discern and select the features for different tasks. Without loss of generality, the linear classifier is employed due to its simplicity and versatility. Specifically, given a feature vector $\boldsymbol{f}$, the predicted logit for class j is computed as $z_j = \boldsymbol{w}_j^\top \boldsymbol{f} + b$. Note that when training with long-tail data, the norms of classifier weight $\boldsymbol{w}_j$ tend to exhibit an imbalanced distribution, which can lead to biased predictions (Kang et al., 2020; Wei et al., 2021). To overcome it and draw on the strengths of CLIP which optimizes cosine distances within its feature space, inspired by Wei et al. (2021), we propose to use a enhanced cosine classifier $z_j = \sigma \cdot \frac{\boldsymbol{w}_j^\top \boldsymbol{f}}{\|\boldsymbol{w}_j\|_2 \|\boldsymbol{f}\|_2}$. Here, σ is a scaling factor, and the biased prediction is alleviated by dividing the classifier norm. To further improve the effectiveness of LIFT, we opt for the LA loss for optimization:

$$\mathcal{L}_{\text{LA}}(\boldsymbol{x}, y = j) = -\log \frac{\exp(z_j + \log \mathrm{P}(y = j))}{\sum_{k \in [K]} \exp(z_k + \log \mathrm{P}(y = k))} \tag{5}$$

Here, $y = j$ represents the ground-truth label of $\boldsymbol{x}$ and z_j is the predicted logit. $\mathrm{P}(y = j)$ signifies the class prior probability, which can be estimated based on the training data. More theoretical understanding of the LA loss can be found in Appendix B.

4.3. Semantic-Aware Initialization

Note that an uninitialized classifier is discovered to have a negative impact on fine-tuning the model (Kumar et al., 2022). Therefore, it is crucial to set an appropriate initial state for the classifier. A straightforward method is to apply linear probing using re-weighted or LA loss. Another approach is to compute the class mean feature as initialization. However, these two approaches not only require extracting features of training data but also are not available with scarce tail-class data. To overcome it, we tend to leverage the semantic knowledge from the text modality of CLIP. Specifically, we generate hand-crafted textual prompts (*e.g.,* "`a photo of a [CLASS].`") and compute their features $\boldsymbol{f}_{T_1}, \cdots, \boldsymbol{f}_{T_K}$, which are then employed to initialize the classfier weights $\boldsymbol{w}_1, \cdots, \boldsymbol{w}_K$. We call this way as *semantic-aware initialization* (SAI). Unlike prior methods that fine-tune both the image encoder and the text encoder in optimization processes (Ma et al., 2021; Tian et al., 2022), SAI relies on a single forward pass of the text encoder for each class description. After that, the text encoder is discarded. This simple approach allows us to achieve a better initial state of the classifier with small computational overhead.

4.4. Test-Time Ensembling

It is well-established that applying random perturbations to each input can lead to improved generalization (Sun et al., 2020; Wang et al., 2021a; Zhou et al., 2023). This principle may also be useful for the Transformer-based foundation model, where an image is divided into multiple patches, potentially resulting in the segmentation of continuous patterns into different patches. To further enhance the generalization robustness, we propose to aggregate the predictions from a set of perturbed versions of the input. Formally, given a test data point $\boldsymbol{x}$, its predicted logits $\boldsymbol{z}$ are obtained by averaging the predictions from M perturbed versions:

$$\boldsymbol{z} = \log \mathrm{P}(\boldsymbol{y} \mid \boldsymbol{x}) = \frac{1}{M} \sum_{i=1}^{M} \log \mathrm{P}(\boldsymbol{y} \mid \alpha_i(\boldsymbol{x})) \tag{6}$$

Here, $\alpha_i(\boldsymbol{x})$ represents different perturbed versions of $\boldsymbol{x}$. This approach helps mitigate bias introduced by image cropping. Due to the page limit, we present the algorithm procedure in Appendix E. We term this technique *test-time ensembling* (TTE), and it can be integrated with small computational overhead to enhance performance.

5. Empirical Study

5.1. Experimental Settings

We conduct experiments on four long-tail datasets, including ImageNet-LT (Liu et al., 2019), Places-LT (Liu et al., 2019), iNaturalist 2018 (Van Horn et al., 2018) and CIFAR-100-LT (Cao et al., 2019). ImageNet-LT has 115.8K images from 1000 classes, with a maximum of 1280 and a minimum of 5 images per class. Places-LT contains 62.5K images from 365 classes, from a maximal 4980 to a minimum of 5 images per class. iNaturalist 2018 consists of 437.5K images distributed across 8142 species, with the number of images per species varying from as few as 2 to as many as 1000. CIFAR-100-LT is constructed with various imbalance ratios, including 100, 50, and 10. In addition to measuring overall accuracy, we adhere to the evaluation protocol introduced by Liu et al. (2019) to report accuracy across three splits of classes: head classes (>100 images), medium classes (20∼100 images), and tail classes (<20 images).

5.2. Comparison with State-of-the-art Methods

Results on ImageNet-LT. We report the test accuracy in Table 1. While existing approaches such as VL-LTR (Tian et al., 2022) and GML (Suh & Seo, 2023) rely on extensive auxiliary data to facilitate fine-tuning, our method LIFT achieves superior performance by leveraging the test-time ensembling (TTE) technique alone. The use of external data not only incurs significant computational overhead but also reduces practicality due to the unavailability of such data in many real-world applications. The advantage of LIFT

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

Methods	Backbone	Learnable Params.	#Epochs	Overall	Head	Medium	Tail
Training from scratch							
cRT (Kang et al., 2020)	ResNet-50	23.51M	90+10	47.3	58.8	44.0	26.1
LWS (Kang et al., 2020)	ResNet-50	23.51M	90+10	47.7	57.1	45.2	29.3
MiSLAS (Zhong et al., 2021)	ResNet-50	23.51M	180+10	52.7	62.9	50.7	34.3
LA (Menon et al., 2021)	ResNet-50	23.51M	90	51.1	-	-	-
DisAlign (Zhang et al., 2021)	ResNet-50	23.51M	90	52.9	61.3	52.2	31.4
BCL (Zhu et al., 2022)	ResNet-50	23.51M	100	56.0	-	-	-
PaCo (Cui et al., 2021)	ResNet-50	23.51M	400	57.0	-	-	-
NCL (Li et al., 2022a)	ResNet-50	23.51M	400	57.4	-	-	-
LiVT (Xu et al., 2023)	ViT-B/16	85.80M	100	60.9	73.6	56.4	41.0
Fine-tuning foundation model							
BALLAD (Ma et al., 2021)	ViT-B/16	149.62M	50+10	75.7	79.1	74.5	69.8
Decoder (Wang et al., 2024)	ViT-B/16	21.26M	∼18	73.2	-	-	-
LIFT (Ours)	ViT-B/16	**0.62M**	**10**	77.0	80.2	76.1	71.5
LIFT w/ TTE (Ours)	ViT-B/16	**0.62M**	**10**	**78.3**	**81.3**	**77.4**	**73.4**
Fine-tuning with extra data							
VL-LTR (Tian et al., 2022)	ViT-B/16	149.62M	100	77.2	84.5	74.6	59.3
GML (Suh & Seo, 2023)	ViT-B/16	149.62M	100	78.0	-	-	-

Table 2: Comparison with state-of-the-art methods on Places-LT.

Methods	Backbone	Learnable Params.	#Epochs	Overall	Head	Medium	Tail
Training from scratch (with an ImageNet-1K pre-trained backbone)							
OLTR (Liu et al., 2019)	ResNet-152	58.14M	30	35.9	44.7	37.0	25.3
cRT (Kang et al., 2020)	ResNet-152	58.14M	90+10	36.7	42.0	37.6	24.9
LWS (Kang et al., 2020)	ResNet-152	58.14M	90+10	37.6	40.6	39.1	28.6
MiSLAS (Zhong et al., 2021)	ResNet-152	58.14M	90+10	40.4	39.6	43.3	36.1
DisAlign (Zhang et al., 2021)	ResNet-152	58.14M	30	39.3	40.4	42.4	30.1
ALA (Zhao et al., 2022)	ResNet-152	58.14M	30	40.1	43.9	40.1	32.9
PaCo (Cui et al., 2021)	ResNet-152	58.14M	30	41.2	36.1	47.9	35.3
LiVT (Xu et al., 2023)	ViT-B/16	85.80M	100	40.8	48.1	40.6	27.5
Fine-tuning foundation model							
BALLAD (Ma et al., 2021)	ViT-B/16	149.62M	50+10	49.5	49.3	50.2	48.4
Decoder (Wang et al., 2024)	ViT-B/16	21.26M	∼34	46.8	-	-	-
LPT (Dong et al., 2023)	ViT-B/16	1.01M	40+40	50.1	49.3	52.3	46.9
LIFT (Ours)	ViT-B/16	**0.18M**	**10**	51.5	51.3	52.2	50.5
LIFT w/ TTE (Ours)	ViT-B/16	**0.18M**	**10**	**52.2**	**51.7**	**53.1**	**50.9**
Fine-tuning with extra data							
VL-LTR (Tian et al., 2022)	ViT-B/16	149.62M	100	50.1	54.2	48.5	42.0
RAC (Long et al., 2022)	ViT-B/16	85.80M	30	47.2	48.7	48.3	41.8

is more significant compared with methods that do not use auxiliary data, *i.e.,* LIFT surpasses the previous best method by 1.3% in accuracy. Importantly, LIFT only needs 10 epochs of training and fine-tunes far fewer model parameters (*i.e.,* from 21.26M to 0.62M). It is worth noting that we do not include LPT (Dong et al., 2023) for comparison since it is pre-trained on ImageNet-21K, its results on ImageNet-LT were not reported in the original paper.

Results on Places-LT. Table 2 shows that LIFT outperforms existing methods by a larger margin on Places-LT. Even without TTE, LIFT surpasses VL-LTR and RAC which use external training data by 1.4% in accuracy. By integrating TTE, the number increases to 2.1%. Similar to ImageNet-LT, we only need 10 epochs of training in contrast to 80 epochs (40 in each stage) for LPT. The amount of tunable parameters is also much fewer, *i.e.,* 0.18M. Never-

Table 3: Comparison with state-of-the-art methods on iNaturalist 2018.

Methods	Backbone	Learnable Params.	#Epochs	Overall	Head	Medium	Tail
Training from scratch							
cRT (Kang et al., 2020)	ResNet-50	23.51M	90+10	65.2	69.0	66.0	63.2
LWS (Kang et al., 2020)	ResNet-50	23.51M	90+10	65.9	65.0	66.3	65.5
MiSLAS (Zhong et al., 2021)	ResNet-50	23.51M	200+30	71.6	73.2	72.4	70.4
DiVE (He et al., 2021)	ResNet-50	23.51M	90	69.1	70.6	70.0	67.6
DisAlign (Zhang et al., 2021)	ResNet-50	23.51M	90	69.5	61.6	70.8	69.9
ALA (Zhao et al., 2022)	ResNet-50	23.51M	90	70.7	71.3	70.8	70.4
RIDE (Wang et al., 2021c)	ResNet-50	23.51M	100	72.6	70.9	72.4	73.1
RIDE+CR (Ma et al., 2023)	ResNet-50	23.51M	200	73.5	71.0	73.8	74.3
RIDE+OTmix (Gao et al., 2023)	ResNet-50	23.51M	210	73.0	71.3	72.8	73.8
BCL (Zhu et al., 2022)	ResNet-50	23.51M	100	71.8	-	-	-
PaCo (Cui et al., 2021)	ResNet-50	23.51M	400	73.2	70.4	72.8	73.6
NCL (Li et al., 2022a)	ResNet-50	23.51M	400	74.2	72.0	74.9	73.8
GML (Suh & Seo, 2023)	ResNet-50	23.51M	400	74.5	-	-	-
LiVT (Xu et al., 2023)	ViT-B/16	85.80M	100	76.1	**78.9**	76.5	74.8
Fine-tuning foundation model							
Decoder (Wang et al., 2024)	ViT-B/16	21.26M	∼**5**	59.2	-	-	-
LPT (Dong et al., 2023)	ViT-B/16	**1.01M**	80+80	76.1	-	-	79.3
LIFT (Ours)	ViT-B/16	4.75M	20	79.1	72.4	79.0	81.1
LIFT w/ TTE (Ours)	ViT-B/16	4.75M	20	**80.4**	74.0	**80.3**	**82.2**
Fine-tuning with extra data							
VL-LTR (Tian et al., 2022)	ViT-B/16	149.62M	100	76.8	-	-	-
RAC (Long et al., 2022)	ViT-B/16	85.80M	20	80.2	75.9	80.5	81.1

theless, LIFT outperforms LPT by 1.4% in accuracy. When taking a closer look, we can see that LIFT significantly improves the tail class performance, *i.e.,* from 46.9 to 50.5.

Results on iNaturalist 2018. We report the results in Table 3. Overall, our method achieves the best performance on this challenging dataset, surpassing LPT, VL-LTR, and RAC. We acknowledge that Decoder (Wang et al., 2024) uses fewer training epochs, however, its performance trails far behind LIFT. Particularly, LIFT (without using TTE) improves LPT by 3% in accuracy and LIFT only needs 20 epochs of training compared with 160 epochs (80 per stage) for LPT. Although LPT uses fewer learnable parameters, we can reduce the parameters of LIFT to reach a comparable quantity (*i.e.,* reduce the bottleneck dimension r to 64, more details are given in Figure 6). In this case, LIFT achieves an accuracy of 77.7% (without TTE) / 79.0% (with TTE), which still outperforms LPT. In fact, due to the large number of classes of iNaturalist 2018, the classifier already contains 6.25M parameters. Therefore, the parameter quantity of LIFT does not lead to too much overhead.

Results on CIFAR-100-LT. The results in Table 4 demonstrate that LIFT outperforms other methods, including LiVT, BALLAD, and various training-from-scratch approaches. These results hold regardless of whether TTE is applied or not. Additionally, we extend our experiments by replacing the CLIP with the pre-trained model from ImageNet-21K. In this case, we employ the class mean features to initialize the classifier due to the lack of a corresponding text encoder. Despite the inherent class overlaps between ImageNet-21K and CIFAR-100, which naturally lead to higher performance, our method surpasses LPT with fewer training epochs and learnable parameters.

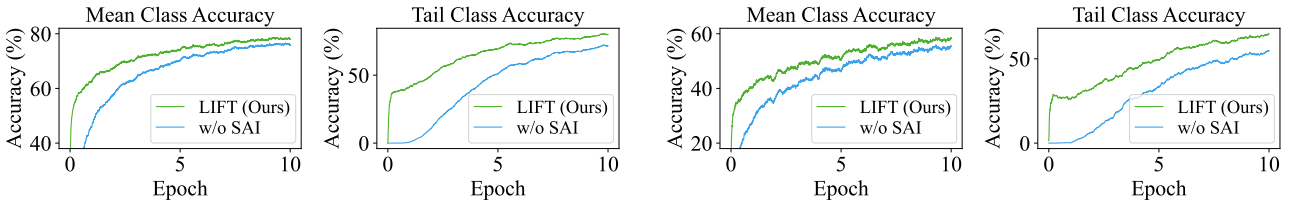
5.3. More Advantages and Ablation Studies

LIFT Improves Convergence. Figure 5 presents the mean class and tail class training accuracy as a function of epochs. Overall, we can observe that LIFT converges rapidly with 10 training epochs on both datasets. As expected, semantic-aware initialization (SAI) attributes to fast convergence, especially in the case of the tail classes.

Ablation Studies. To assess the impact of each component, we conduct a systematical ablation study on different components in LIFT including 1) the structured lightweight fine-tuning (SLF) module, 2) the logit-adjusted (LA) loss, 3) semantic-aware initialization (SAI), and 4) test-time ensembling (TTE). The results presented in Table 5 demonstrate the effectiveness of each component. Specifically, 1) SLF enhances performance on both head and tail classes; 2) without the LA loss, predictions tend to be biased to head classes; 3) SAI consistently improves the generalization, particularly

Table 4: Comparison with state-of-the-art methods on CIFAR-100-LT with various imbalance ratios. †Pre-trained model from ImageNet-21K[1] has several classes related to CIFAR-100[2], which potentially leads to data leakage.

Methods	Backbone	Learnable Params.	#Epochs	Imbalance Ratio 100	50	10
Training from scratch						
LDAM (Cao et al., 2019)	ResNet-32	0.46M	200	42.0	46.6	58.7
BBN (Zhou et al., 2020)	ResNet-32	0.46M	200	42.6	47.0	59.1
DiVE (He et al., 2021)	ResNet-32	0.46M	200	45.4	51.1	62.0
MiSLAS (Zhong et al., 2021)	ResNet-32	0.46M	200+10	47.0	52.3	63.2
BS (Ren et al., 2020)	ResNet-32	0.46M	400	50.8	54.2	63.0
PaCo (Cui et al., 2021)	ResNet-32	0.46M	400	52.0	56.0	64.2
BCL (Zhu et al., 2022)	ResNet-32	0.46M	200	51.9	56.6	64.9
Fine-tuning pre-trained model						
LiVT (Xu et al., 2023)	ViT-B/16	85.80M	100	58.2	-	69.2
BALLAD (Ma et al., 2021)	ViT-B/16	149.62M	50+10	77.8	-	-
LIFT (Ours)	ViT-B/16	**0.10M**	**10**	80.3	82.0	83.8
LIFT w/ TTE (Ours)	ViT-B/16	**0.10M**	**10**	**81.7**	**83.1**	**84.9**
Fine-tuning pre-trained model from ImageNet-21K†						
LPT (Dong et al., 2023)	ViT-B/16	1.01M	40+40	**89.1**	90.0	91.0
LIFT (Ours)	ViT-B/16	**0.10M**	**10**	**89.1**	**90.2**	**91.3**



(a) ImageNet-LT. (b) Places-LT.

Figure 5: Convergence curve of mean class and tail class training accuracy.

Table 5: Ablation study of each key component in LIFT. The baseline involves learning a cosine classifier using CE loss.

SLF	LA	SAI	TTE	ImageNet-LT Overall	Head	Medium	Tail	Places-LT Overall	Head	Medium	Tail
				60.9	82.6	56.7	13.8	34.3	53.6	30.5	7.5
✓				68.9	**84.7**	66.3	33.7	38.9	**55.6**	35.3	16.4
✓	✓			74.9	79.7	74.7	61.7	48.7	50.6	50.8	40.5
✓	✓	✓		77.0	80.2	76.1	71.5	51.5	51.3	52.2	50.5
✓	✓	✓	✓	**78.3**	81.3	**77.4**	**73.4**	**52.2**	51.7	**53.1**	**50.9**

on tail classes; 4) TTE further boosts the performance across both head and tail classes.

Combined with Varying Fine-Tuning Methods. LIFT is a general framework in which many lightweight fine-tuning methods can be integrated. In addition to zero-shot CLIP, classifier fine-tuning, and full fine-tuning, we test LIFT with arbitrary lightweight fine-tuning and other 6 structured lightweight methods, including *BitFit* (Zaken et al., 2022), *VPT-shallow* (Jia et al., 2022), *VPT-deep* (Jia et al., 2022), *Adapter* (Houlsby et al., 2019), *LoRA* (Hu et al., 2022), and *AdaptFormer* (Chen et al., 2022). The results in Table 6 demonstrate that arbitrary lightweight fine-tuning surpasses the baseline methods by a large margin across both overall and tail-class performance. This underscores the efficacy of lightweight fine-tuning even in the absence of specific strategies. Furthermore, the integration of all structured methods leads to improved performance. Specifically, AdaptFormer performs best on both datasets and Adapter achieves slightly low accuracy (*i.e.,* 0.2%).

[1] https://storage.googleapis.com/bit_models/imagenet21k_wordnet_lemmas.txt

[2] https://www.cs.toronto.edu/%7Ekriz/cifar.html

Table 6: Comparison of different fine-tuning methods. All methods use SAI (if have classifier) and TTE for fair comparison.

Methods		ImageNet-LT				Places-LT			
		Overall	Head	Medium	Tail	Overall	Head	Medium	Tail
Zero-shot CLIP		68.3	69.2	67.6	67.7	40.2	38.3	39.2	45.9
Classifier fine-tuning		74.4	77.6	73.7	68.0	49.4	49.1	50.3	48.2
Full fine-tuning		74.4	**82.2**	73.9	53.8	47.2	51.6	48.5	36.2
LIFT (Arbitrary)		77.8	80.9	76.9	72.2	51.9	51.3	52.7	**51.1**
LIFT (Structured)	BitFit	77.0	79.7	76.3	71.9	51.5	51.2	52.3	50.0
	VPT-shallow	75.2	78.8	74.8	66.8	49.9	50.5	51.4	45.3
	VPT-deep	77.2	79.5	76.5	72.8	51.5	51.4	52.3	49.8
	Adapter	78.1	81.3	77.1	72.8	52.0	**51.7**	52.7	51.0
	LoRA	76.9	79.6	76.2	71.7	51.8	51.5	52.5	50.5
	AdaptFormer	**78.3**	81.3	**77.4**	**73.4**	**52.2**	**51.7**	**53.1**	50.9

Table 7: Comparison of LIFT with different classifier initialization methods.

Methods	ImageNet-LT				Places-LT			
	Overall	Head	Medium	Tail	Overall	Head	Medium	Tail
Random initialization	76.1	80.8	75.9	63.2	49.6	51.2	52.1	41.0
Linear probing (re-weighted)	77.1	**81.8**	76.4	66.6	49.9	51.2	51.0	45.0
Linear probing (w/ LA loss)	77.2	81.3	76.4	68.4	50.2	51.3	51.1	46.2
Class mean features	77.5	81.3	76.8	69.4	51.3	51.6	52.1	48.8
Semantic-aware initialization	**78.3**	81.3	**77.4**	**73.4**	**52.2**	**51.7**	**53.1**	**50.9**

Effect of Semantic-Aware Initialization. In Table 7, we test three kinds of classifier initialization strategies in comparison with the random initialization baseline. In LIFT, we use the textual feature by default because it transfers semantic relations between classes during fine-tuning. We also compare different textual prompting methods and report the results in Appendix F. The other strategies using linear probing or class mean features to initialize the classifier achieve slightly poor performance but still significantly improve the baseline. This experiment shows that a good starting point for parameter optimization can lead to a better solution.

5.4. More Supporting Experiments

Due to the page limitation, we report additional results in Appendix G, including 1) comparison of different training epochs; 2) comparison of different classifiers; 3) comparison of different losses; 4) more detailed observations on model convergence; 5) LIFT with variant backbones.

6. Conclusion

This paper studies long-tail learning with the foundation model. We analyze the issue of heavy fine-tuning in distorting tail-class performance and discover that even an arbitrary lightweight fine-tuning yields significant improvement. Moreover, we propose a versatile framework LIFT to facilitate lightweight fine-tuning for long-tail learning. LIFT notably achieves convergence in fewer than 20 training epochs without the need for any external data, and consistently outperforms numerous baseline methods across a range of long-tail datasets, including CIFAR-100-LT, ImageNet-LT, Places-LT, and iNaturalist 2018. We emphasize the ease of training and hope that our approach serves as an inspiration for further advancements in the field of long-tail learning.

Code Availability Statement

The implementation code for this work is available at https://github.com/shijxcs/LIFT.

Acknowledgements

This research was supported by the National Key R&D Program of China (2022YFC3340901), the National Science Foundation of China (62176118).

Impact Statement

This paper aims to advance the field of long-tail learning. In critical and high-stakes applications such as autonomous driving, medical image diagnosis, and industrial anomaly detection, the presence of long-tail data poses the risk of producing biased predictions. Despite the strong capacity of the foundation model, its deployment may result in non-negligible performance degradation if not utilized properly. By shedding light on this problem, we aim to inspire more research on efficient and accurate long-tail learning algorithms with foundation models.

References

Ahn, S., Ko, J., and Yun, S.-Y. CUDA: Curriculum of data augmentation for long-tailed recognition. In *International Conference on Learning Representations*, 2023.

Ba, J. L., Kiros, J. R., and Hinton, G. E. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Cao, K., Wei, C., Gaidon, A., Arechiga, N., and Ma, T. Learning imbalanced datasets with label-distribution-aware margin loss. In *Advances in Neural Information Processing Systems*, volume 32, pp. 1565–1576, 2019.

Chen, S., GE, C., Tong, Z., Wang, J., Song, Y., Wang, J., and Luo, P. Adaptformer: Adapting vision transformers for scalable visual recognition. In *Advances in Neural Information Processing Systems*, volume 35, pp. 16664–16678, 2022.

Cui, J., Zhong, Z., Liu, S., Yu, B., and Jia, J. Parametric contrastive learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 715–724, 2021.

Cui, Y., Jia, M., Lin, T.-Y., Song, Y., and Belongie, S. Class-balanced loss based on effective number of samples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9268–9277, 2019.

Devlin, J., Chang, M., Lee, K., and Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pp. 4171–4186, 2019.

Dong, B., Zhou, P., Yan, S., and Zuo, W. LPT: Long-tailed prompt tuning for image classification. In *International Conference on Learning Representations*, 2023.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

Gao, J., Zhao, H., Li, Z., and Guo, D. Enhancing minority classes by mixing: An adaptative optimal transport approach for long-tailed classification. In *Advances in Neural Information Processing Systems*, volume 36, pp. 60329–60348, 2023.

Han, B. Wrapped cauchy distributed angular softmax for long-tailed visual recognition. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 12368–12388, 2023.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

He, K., Chen, X., Xie, S., Li, Y., Dollár, P., and Girshick, R. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16000–16009, 2022.

He, X., Fu, S., Ding, X., Cao, Y., and Wang, H. Uniformly distributed category prototype-guided vision-language framework for long-tail recognition. In *Proceedings of the 31st ACM International Conference on Multimedia*, pp. 5027–5037, 2023.

He, Y.-Y., Wu, J., and Wei, X.-S. Distilling virtual examples for long-tailed recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 235–244, 2021.

Hong, Y., Han, S., Choi, K., Seo, S., Kim, B., and Chang, B. Disentangling label distribution for long-tailed visual recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6626–6636, 2021.

Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., Attariyan, M., and Gelly, S. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning*, pp. 2790–2799, 2019.

Hu, E. J., yelong shen, Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Iscen, A., Fathi, A., and Schmid, C. Improving image recognition by retrieving from web-scale image-text data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 19295–19304, 2023.

Jia, L.-H., Guo, L.-Z., Zhou, Z., and Li, Y.-F. LAMDA-SSL: A comprehensive semi-supervised learning toolkit. *SCIENCE CHINA Information Sciences*, 67(1):117101, 2024.

Jia, M., Tang, L., Chen, B.-C., Cardie, C., Belongie, S., Hariharan, B., and Lim, S.-N. Visual prompt tuning. In *Proceedings of the 17th European Conference on Computer Vision*, pp. 709–727, 2022.

Kang, B., Xie, S., Rohrbach, M., Yan, Z., Gordo, A., Feng, J., and Kalantidis, Y. Decoupling representation and classifier for long-tailed recognition. In *International Conference on Learning Representations*, 2020.

Kang, B., Li, Y., Xie, S., Yuan, Z., and Feng, J. Exploring balanced feature spaces for representation learning. In *International Conference on Learning Representations*, 2021.

Kumar, A., Raghunathan, A., Jones, R. M., Ma, T., and Liang, P. Fine-tuning can distort pretrained features and underperform out-of-distribution. In *International Conference on Learning Representations*, 2022.

Li, B., Yao, Y., Tan, J., Gong, R., Lu, J., and Luo, Y. Rectify representation bias in vision-language models for long-tailed recognition. *Neural Networks*, 172:106134, 2024.

Li, J., Tan, Z., Wan, J., Lei, Z., and Guo, G. Nested collaborative learning for long-tailed visual recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6949–6958, 2022a.

Li, M., Cheung, Y.-m., and Lu, Y. Long-tailed visual recognition via gaussian clouded logit adjustment. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6929–6938, 2022b.

Lian, D., Zhou, D., Feng, J., and Wang, X. Scaling & shifting your features: A new baseline for efficient model tuning. In *Advances in Neural Information Processing Systems*, volume 35, pp. 109–123, 2022.

Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P. Focal loss for dense object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2980–2988, 2017.

Liu, Z., Miao, Z., Zhan, X., Wang, J., Gong, B., and Yu, S. X. Large-scale long-tailed recognition in an open world. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2537–2546, 2019.

Long, A., Yin, W., Ajanthan, T., Nguyen, V., Purkait, P., Garg, R., Blair, A., Shen, C., and van den Hengel, A. Retrieval augmented classification for long-tail visual recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6959–6969, 2022.

Ma, T., Geng, S., Wang, M., Shao, J., Lu, J., Li, H., Gao, P., and Qiao, Y. A simple long-tailed recognition baseline via vision-language model. *arXiv preprint arXiv:2111.14745*, 2021.

Ma, Y., Jiao, L., Liu, F., Yang, S., Liu, X., and Li, L. Curvature-balanced feature manifold learning for long-tailed classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15824–15835, 2023.

Menon, A. K., Jayasumana, S., Rawat, A. S., Jain, H., Veit, A., and Kumar, S. Long-tail learning via logit adjustment. In *International Conference on Learning Representations*, 2021.

Menon, S. and Vondrick, C. Visual classification via description from large language models. In *International Conference on Learning Representations*, 2023.

Nam, G., Jang, S., and Lee, J. Decoupled training for long-tailed classification with stochastic representations. In *International Conference on Learning Representations*, 2023.

Park, S., Hong, Y., Heo, B., Yun, S., and Choi, J. Y. The majority can help the minority: Context-rich minority oversampling for long-tailed classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6887–6896, 2022.

Peifeng, G., Xu, Q., Wen, P., Yang, Z., Shao, H., and Huang, Q. Feature directions matter: Long-tailed learning via rotated balanced representation. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 27542–27563, 2023.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning*, pp. 8748–8763, 2021.

Ren, J., Yu, C., sheng, s., Ma, X., Zhao, H., Yi, S., and Li, h. Balanced meta-softmax for long-tailed visual recognition. In *Advances in Neural Information Processing Systems*, volume 33, pp. 4175–4186, 2020.

Samuel, D. and Chechik, G. Distributional robustness loss for long-tail learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9495–9504, 2021.

Shi, J.-X., Wei, T., Xiang, Y., and Li, Y.-F. How re-sampling helps for long-tail learning? In *Advances in Neural Information Processing Systems*, volume 36, pp. 75669–75687, 2023.

Shi, J.-X., Wei, T., and Li, Y.-F. Residual diverse ensemble for long-tailed multi-label text classification. *SCIENCE CHINA Information Sciences*, 2024.

Song, Y., Li, M., and Wang, B. Long-tailed visual recognition via improved cross-window self-attention and trivialaugment. *IEEE Access*, 11:49601–49610, 2023.

Steiner, A., Kolesnikov, A., Zhai, X., Wightman, R., Uszkoreit, J., and Beyer, L. How to train your vit? data, augmentation, and regularization in vision transformers. *arXiv preprint arXiv:2106.10270*, 2021.

Suh, M.-K. and Seo, S.-W. Long-tailed recognition by mutual information maximization between latent features and ground-truth labels. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202, pp. 32770–32782, 2023.

Sun, Y., Wang, X., Liu, Z., Miller, J., Efros, A., and Hardt, M. Test-time training with self-supervision for generalization under distribution shifts. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 9229–9248, 2020.

Tian, C., Wang, W., Zhu, X., Dai, J., and Qiao, Y. VL-LTR: learning class-wise visual-linguistic representation for long-tailed visual recognition. In *Proceedings of the 17th European Conference on Computer Vision*, pp. 73–91, 2022.

Van Horn, G., Mac Aodha, O., Song, Y., Cui, Y., Sun, C., Shepard, A., Adam, H., Perona, P., and Belongie, S. The inaturalist species classification and detection dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8769–8778, 2018.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pp. 5998–6008, 2017.

Wang, D., Shelhamer, E., Liu, S., Olshausen, B., and Darrell, T. Tent: Fully test-time adaptation by entropy minimization. In *International Conference on Learning Representations*, 2021a.

Wang, P., Han, K., Wei, X.-S., Zhang, L., and Wang, L. Contrastive learning based hybrid networks for long-tailed image classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 943–952, 2021b.

Wang, X., Lian, L., Miao, Z., Liu, Z., and Yu, S. Long-tailed recognition by routing diverse distribution-aware experts. In *International Conference on Learning Representations*, 2021c.

Wang, Y., Yu, Z., Wang, J., Heng, Q., Chen, H., Ye, W., Xie, R., Xie, X., and Zhang, S. Exploring vision-language models for imbalanced learning. *International Journal of Computer Vision*, 132:224–237, 2024.

Wei, T. and Gan, K. Towards realistic long-tailed semi-supervised learning: Consistency is all you need. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3469–3478, 2023.

Wei, T., Tu, W.-W., Li, Y.-F., and Yang, G.-P. Towards robust prediction on tail labels. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 1812–1820, 2021.

Wei, T., Wang, H., Tu, W.-W., and Li, Y.-F. Robust model selection for positive and unlabeled learning with constraints. *Science China Information Sciences*, 65(11): 212101, 2022.

Xia, P., Xu, D., Ju, L., Hu, M., Chen, J., and Ge, Z. Lmpt: Prompt tuning with class-specific embedding loss for long-tailed multi-label visual recognition. *arXiv preprint arXiv:2305.04536*, 2023.

Xu, Z., Liu, R., Yang, S., Chai, Z., and Yuan, C. Learning imbalanced data with vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15793–15803, 2023.

Yang, Y. and Xu, Z. Rethinking the value of labels for improving class-imbalanced learning. In *Advances in Neural Information Processing Systems*, volume 33, pp. 19290–19301, 2020.

Yang, Y., Chen, S., Li, X., Xie, L., Lin, Z., and Tao, D. Inducing neural collapse in imbalanced learning: Do we really need a learnable classifier at the end of deep neural network? In *Advances in Neural Information Processing Systems*, volume 35, pp. 37991–38002, 2022.

Yu, B. X., Chang, J., Wang, H., Liu, L., Wang, S., Wang, Z., Lin, J., Xie, L., Li, H., Lin, Z., et al. Visual tuning. *arXiv preprint arXiv:2305.06061*, 2023.

Zaken, E. B., Goldberg, Y., and Ravfogel, S. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 1–9, 2022.

Zhang, S., Li, Z., Yan, S., He, X., and Sun, J. Distribution alignment: A unified framework for long-tail visual recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2361–2370, 2021.

Zhao, Y., Chen, W., Tan, X., Huang, K., and Zhu, J. Adaptive logit adjustment loss for long-tailed visual recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 3472–3480, 2022.

Zhong, Z., Cui, J., Liu, S., and Jia, J. Improving calibration for long-tailed recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16489–16498, 2021.

Zhou, B., Cui, Q., Wei, X.-S., and Chen, Z.-M. BBN: Bilateral-branch network with cumulative learning for long-tailed visual recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9719–9728, 2020.

Zhou, K., Yang, J., Loy, C. C., and Liu, Z. Conditional prompt learning for vision-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16816–16825, 2022a.

Zhou, K., Yang, J., Loy, C. C., and Liu, Z. Learning to prompt for vision-language models. *International Journal of Computer Vision*, 130(9):2337–2348, 2022b.

Zhou, Z., Guo, L.-Z., Jia, L.-H., Zhang, D., and Li, Y.-F. ODS: Test-time adaptation in the presence of open-world data shift. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 42574–42588, 2023.

Zhou, Z., Yang, M., Shi, J.-X., Guo, L.-Z., and Li, Y.-F. Decoop: Robust prompt tuning with out-of-distribution detection. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

Zhu, J., Wang, Z., Chen, J., Chen, Y.-P. P., and Jiang, Y.-G. Balanced contrastive learning for long-tailed visual recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6908–6917, 2022.

A. Implementation Details

For all experiments, we use the SGD optimizer with a batch size of 128, weight decay of 5×10^{-4}, and momentum of 0.9. For lightweight fine-tuning methods, the learning rate is 0.01. For full fine-tuning, we search the learning rate from $\{0.02, 0.01, 0.005, 0.002, 0.001, 0.0005\}$ considering its weak stability. For ImageNet-LT, Places-LT, and CIFAR-100-LT, we train the model for only 10 epochs; and for iNaturalist 2018, we train 20 epochs considering that it has much more data. In LIFT, we set the bottleneck dimension $r = 2^{\lfloor \log_2(\frac{K}{2L}) \rfloor}$ for Adapter and AdaptFormer such that it learns even fewer parameters than the classifier (please refer to Appendix D for detailed analysis). The scaling factor σ of the cosine classifier is set to 25 (please refer to Table 15 and the corresponding paragraph for the analysis). All experiments are conducted on a single NVIDIA A800 GPU. In fact, a GPU with 20GB of memory is sufficient for all reproductions.

B. Understanding Logit-Adjusted Loss

Proof. Logit-adjusted loss (Menon et al., 2021) (or termed Balanced-Softmax (Ren et al., 2020)) is widely used in previous literature (Hong et al., 2021; Zhao et al., 2022; Li et al., 2022b) because of its theoretical optimality and formal simplicity. Following we give a brief proof:

$$\begin{aligned}
\mathrm{P}_s(y=j \mid \boldsymbol{x}) &= \frac{\mathrm{P}_s(y=j \mid \boldsymbol{x})}{\sum_{k\in[K]} \mathrm{P}_s(y=k \mid \boldsymbol{x})} \\
&= \frac{\mathrm{P}_t(y=j \mid \boldsymbol{x}) \cdot \frac{\mathrm{P}_s(y=j \mid \boldsymbol{x})}{\mathrm{P}_t(y=j \mid \boldsymbol{x})}}{\sum_{k\in[K]} \mathrm{P}_t(y=k \mid \boldsymbol{x}) \cdot \frac{\mathrm{P}_s(y=k \mid \boldsymbol{x})}{\mathrm{P}_t(y=k \mid \boldsymbol{x})}} \\
&= \frac{\mathrm{P}_t(y=j \mid \boldsymbol{x}) \cdot \frac{\mathrm{P}_s(\boldsymbol{x} \mid y=j)}{\mathrm{P}_t(\boldsymbol{x} \mid y=j)} \cdot \frac{\mathrm{P}_s(y=j)}{\mathrm{P}_t(y=j)} \cdot \frac{\mathrm{P}_t(\boldsymbol{x})}{\mathrm{P}_s(\boldsymbol{x})}}{\sum_{k\in[K]} \mathrm{P}_t(y=k \mid \boldsymbol{x}) \cdot \frac{\mathrm{P}_s(\boldsymbol{x} \mid y=k)}{\mathrm{P}_t(\boldsymbol{x} \mid y=k)} \cdot \frac{\mathrm{P}_s(y=k)}{\mathrm{P}_t(y=k)} \cdot \frac{\mathrm{P}_t(\boldsymbol{x})}{\mathrm{P}_s(\boldsymbol{x})}} \\
&= \frac{\mathrm{P}_t(y=j \mid \boldsymbol{x}) \cdot \frac{\mathrm{P}_s(\boldsymbol{x} \mid y=j)}{\mathrm{P}_t(\boldsymbol{x} \mid y=j)} \cdot \frac{\mathrm{P}_s(y=j)}{\mathrm{P}_t(y=j)}}{\sum_{k\in[K]} \mathrm{P}_t(y=k \mid \boldsymbol{x}) \cdot \frac{\mathrm{P}_s(\boldsymbol{x} \mid y=k)}{\mathrm{P}_t(\boldsymbol{x} \mid y=k)} \cdot \frac{\mathrm{P}_s(y=k)}{\mathrm{P}_t(y=k)}}
\end{aligned} \tag{7}$$

where P_s is the probability distribution in the source domain (*i.e.,* the training dataset) and P_t is the probability distribution in the target domain (*i.e.,* the test dataset). For long-tail learning, $\mathrm{P}_s(y)$ appears a long-tail distribution and $\mathrm{P}_t(y)$ is a uniform distribution, *i.e.,* $\mathrm{P}_t(y=k) = \frac{1}{K}$. Moreover, by assuming that $\mathrm{P}_s(\boldsymbol{x} \mid y) = \mathrm{P}_t(\boldsymbol{x} \mid y)$, we have

$$\mathrm{P}_s(y=j \mid \boldsymbol{x}) = \frac{\mathrm{P}_t(y=j \mid \boldsymbol{x}) \cdot \mathrm{P}_s(y=j)}{\sum_{k\in[K]} \mathrm{P}_t(y=k \mid \boldsymbol{x}) \cdot \mathrm{P}_s(y=k)} \tag{8}$$

In deep classification models, $\mathrm{P}_t(y \mid \boldsymbol{x})$ is estimated by the Softmax of the output logits $\boldsymbol{z}$, which is

$$\mathrm{P}_t(y=j \mid \boldsymbol{x}) = \frac{\exp(z_j)}{\sum_{k\in[K]} \exp(z_k)} \tag{9}$$

In order to get the optimal probability model in the target domain, we substitute $\mathrm{P}_t(y \mid \boldsymbol{x})$ in Equation (8) with Equation (9). In this way, we optimize the predicted probability in the source domain as

$$\mathrm{P}_s(y=j \mid \boldsymbol{x}) = \frac{\frac{\exp(z_j)}{\sum_{k\in[K]} \exp(z_k)} \cdot \mathrm{P}_s(y=j)}{\sum_{k\in[K]} \frac{\exp(z_k)}{\sum_{k'\in[K]} \exp(z'_k)} \cdot \mathrm{P}_s(y=k)}$$

$$= \frac{\exp(z_j) \cdot \mathrm{P}_s(y=j)}{\sum_{k\in[K]} \exp(z_k) \cdot \mathrm{P}_s(y=k)}$$
$$= \frac{\exp\left(z_j + \log \mathrm{P}_s(y=j)\right)}{\sum_{k\in[K]} \exp\left(z_k + \log \mathrm{P}_s(y=k)\right)} \tag{10}$$

Therefore, the logit-adjusted loss is defined as

$$\mathcal{L}_{\mathrm{LA}}(\boldsymbol{x}, y=j) = -\log \mathrm{P}_s(y=j \mid \boldsymbol{x}) = -\log \frac{\exp(z_j + \log \mathrm{P}_s(y=j))}{\sum_{k\in[K]} \exp(z_k + \log \mathrm{P}_s(y=k))} \tag{11}$$

In practice, $\mathrm{P}_s(y)$ can be estimated by calculating the class frequency in the training dataset. Compared to other elaborately designed losses, the LA loss does not require any hyperparameters and has fewer assumptions about the data distribution, making it more generalizable across different backbone models. As shown in Table 16, some other elaborately designed losses such as LDAM and LADE, perform unsatisfactorily when applied to the CLIP model.

It is worth mentioning that, there is also a post-hoc logit-adjustment method, where $\mathrm{P}_s(y=j \mid \boldsymbol{x})$ is directly calculated by the Softmax of the logits $\boldsymbol{z}$. In this case, the posterior probability in the target domain can be estimated by

$$\mathrm{P}_t(y=j \mid \boldsymbol{x}) = \frac{\exp\left(z_j - \log \mathrm{P}_s(y=j)\right)}{\sum_{k\in[K]} \exp\left(z_k - \log \mathrm{P}_s(y=k)\right)} \tag{12}$$

The proof is similar to the above.

Impact of Class-Conditional Distribution. Logit-adjusted loss assumes that the class-conditional distribution is consistent between the source domain (the training data) and the target domain (the test data), *i.e.,* $\mathrm{P}_s(\boldsymbol{x} \mid y) = \mathrm{P}_t(\boldsymbol{x} \mid y)$. However, our empirical findings in Figure 3c reveal that full fine-tuning breaks this assumption. By optimizing all of the model parameters, the inter-class similarities and the intra-class distances are significantly reduced. However, the limited data in tail classes makes it hard to generalize in the unknown test scenarios, and the intra-class distances of tail classes in test data remain at a high level. In this case, using P_s to estimate the unknown data $\boldsymbol{x}$ in P_t will lead to an underestimated $\mathrm{P}_t(\boldsymbol{x} \mid y)$.

By revisiting the proof of logit-adjusted loss from Equations (7) to (11) and removing the consistency assumption for class-conditional distribution, we have

$$\begin{aligned}\mathcal{L}_{\mathrm{LA}}(\boldsymbol{x}, y=j) = -\log \mathrm{P}_s(y=j \mid \boldsymbol{x}) &= -\log \frac{\mathrm{P}_t(y=j \mid \boldsymbol{x}) \cdot \mathrm{P}_s(y=j) \cdot \zeta_{s-t}(j)}{\sum_{k\in[K]} \mathrm{P}_t(y=k \mid \boldsymbol{x}) \cdot \mathrm{P}_s(y=k) \cdot \zeta_{s-t}(k)} \\ &= -\log \frac{\exp\left(z_j + \log \mathrm{P}_s(y=j) + \log \zeta_{s-t}(j)\right)}{\sum_{k\in[K]} \exp\left(z_k + \log \mathrm{P}_s(y=k) + \log \zeta_{s-t}(k)\right)}\end{aligned} \tag{13}$$

where $\zeta_{s-t}(k)$ denotes $\dfrac{\mathrm{P}_s(\boldsymbol{x} \mid y=k)}{\mathrm{P}_t(\boldsymbol{x} \mid y=k)}$. For samples of class j, the underestimate of $\mathrm{P}_t(\boldsymbol{x} \mid y=j)$ results in the underestimate of $\mathcal{L}_{\mathrm{LA}}(\boldsymbol{x}, y=j)$, thus inevitably leading to the optimization biased towards other classes and the performance decline of class j. Similarly, for post-hoc logit adjustment in Equation (12), the estimated posterior probability should be

$$\mathrm{P}_t(y=j \mid \boldsymbol{x}) = \frac{\exp\left(z_j - \log \mathrm{P}_s(y=j) - \log \zeta_{s-t}(j)\right)}{\sum_{k\in[K]} \exp\left(z_k - \log \mathrm{P}_s(y=k) - \log \zeta_{s-t}(k)\right)} \tag{14}$$

and the underestimate of $\mathrm{P}_t(\boldsymbol{x} \mid y=j)$ will result in the underestimate of posterior probability $\mathrm{P}_t(y=j \mid \boldsymbol{x})$.

Since the class-conditional distribution in the target domain is hard to acquire, it is impractical to estimate the precise posterior probability given a distorted representation. Nevertheless, given that the foundation model has a strong representation learning capability, we can improve its separability as well as preserve its class-conditional distribution. Our empirical studies in Figure 4 demonstrate that fine-tuning a small portion of the parameters can achieve such effects.

C. Quantifying the Overfitting Issue of Full Fine-Tuning

To give a more thorough comparison between full fine-tuning, classifier fine-tuning, and LIFT, we compute their training and test accuracy, as well as the accuracy gap, and report the results in Table 8. We highlight the overfitting results with red colors. The results show that full fine-tuning tends to cause overfitting, as the gaps between training and test accuracy are obviously higher than the other two methods, particularly on the tail classes.

Table 8: Training and test accuracy of different fine-tuning methods on ImageNet-LT, Places-LT, and iNaturalist 2018. Note that for the class-imbalanced training data, we calculate their mean class accuracy. The red number denotes that the gap between training and test accuracy is larger than LIFT by at least 1%, which indicates overfitting.

(a) ImageNet-LT.

Methods	**Overall**			**Head**			**Medium**			**Tail**		
	train	test	Δ	train	test	Δ	train	test	Δ	train	test	Δ
Full fine-tuning (*best lr*)	91.6	72.9	18.7	92.3	80.8	11.5	88.5	72.4	16.1	72.8	52.1	20.7
Full fine-tuning (*lr equal to LIFT*)	91.6	61.7	29.9	91.8	70.3	21.5	91.3	60.0	21.3	86.0	43.4	42.6
Classifier (*best lr*)	86.1	73.5	12.6	83.2	76.6	6.6	86.9	72.8	14.1	91.7	67.2	24.5
Classifier (*lr equal to LIFT*)	82.8	73.1	9.7	82.6	77.0	5.6	83.8	73.1	10.7	79.3	61.6	17.7
LIFT (Ours)	87.1	77.0	10.1	86.8	80.2	6.6	87.9	76.1	11.8	88.9	71.5	17.4

(b) Places-LT.

Methods	**Overall**			**Head**			**Medium**			**Tail**		
	train	test	Δ	train	test	Δ	train	test	Δ	train	test	Δ
Full fine-tuning (*best lr*)	74.7	46.4	28.3	74.8	51.0	23.8	78.1	47.6	30.5	66.8	35.3	31.5
Full fine-tuning (*lr equal to LIFT*)	81.9	41.8	40.1	77.0	46.3	30.7	85.7	43.1	42.6	82.0	30.5	51.5
Classifier (*best lr*)	64.8	48.5	16.3	55.0	48.6	6.4	66.3	49.1	17.2	79.1	46.8	32.3
Classifier (*lr equal to LIFT*)	59.6	48.3	11.3	54.5	49.3	5.2	62.4	49.9	12.5	62.6	42.5	20.1
LIFT (Ours)	64.2	51.5	12.7	58.3	51.3	7.0	65.9	52.2	13.7	71.1	50.5	20.6

(c) iNaturalist 2018.

Methods	**Overall**			**Head**			**Medium**			**Tail**		
	train	test	Δ	train	test	Δ	train	test	Δ	train	test	Δ
Full fine-tuning (*best lr*)	95.9	72.7	23.2	91.8	70.3	21.5	96.1	73.1	23.0	96.8	72.7	24.1
Full fine-tuning (*lr equal to LIFT*)	95.8	70.1	25.7	85.1	63.7	21.4	95.8	70.0	25.8	98.7	71.8	26.9
Classifier (*best lr*)	76.8	57.9	18.9	53.7	46.7	7.0	74.7	56.8	17.9	85.4	62.1	23.3
Classifier (*lr equal to LIFT*)	76.8	57.9	18.9	53.7	46.7	7.0	74.7	56.8	17.9	85.4	62.1	23.3
LIFT (Ours)	97.3	79.1	18.2	88.0	72.4	15.6	97.4	79.0	18.4	99.4	81.1	18.3

D. Detailed Analysis of Structured Lightweight Fine-Tuning

Preliminary to the Transformer Architecture. A Transformer model consists of an embedding layer and multiple Transformer blocks. Formally, it first divides an input image $\boldsymbol{x}$ into m patches $\{\boldsymbol{x}_i^{\mathrm{p}}\}_{i=1}^m$. These patches are then embeded into sequences of d-dimensional vectors $\boldsymbol{E}^0 = \mathrm{Embed}([\boldsymbol{x}_1^{\mathrm{p}}; \cdots; \boldsymbol{x}_m^{\mathrm{p}}]) \in \mathbb{R}^{m \times d}$. The input embeddings are subsequently passed through L Transformer blocks $\{\Phi^l\}_{l=1}^L$ within the model:

$$\boldsymbol{X}^l = \Phi^l(\boldsymbol{X}^{l-1}). \text{ Specifically, } \begin{cases} \hat{\boldsymbol{X}}^l = \mathrm{MSA}(\mathrm{LN}(\boldsymbol{X}^{l-1})) + \boldsymbol{X}^{l-1} \\ \boldsymbol{X}^l = \mathrm{FFN}(\mathrm{LN}(\hat{\boldsymbol{X}}^l)) + \hat{\boldsymbol{X}}^l \end{cases} \tag{15}$$

$$\mathrm{MSA}^{(l)}(\boldsymbol{X}) = \mathrm{Concat}_{h=1}^H \left(\mathrm{Softmax}\left(\frac{\boldsymbol{X}\boldsymbol{W}_Q^{l,h}(\boldsymbol{X}\boldsymbol{W}_K^{l,h})^\top}{\sqrt{d}} \right) \boldsymbol{X}\boldsymbol{W}_V^{l,h} \right) \boldsymbol{W}_O^l \tag{16}$$

$$\mathrm{FFN}^{(l)}(\boldsymbol{X}) = \mathrm{ReLU}(\boldsymbol{X}\boldsymbol{W}_1^l)\boldsymbol{W}_2^l \tag{17}$$

Here, MSA denotes the multi-head self-attention and H is the number of heads. FFN indicates the feed-forward network, and LN denotes layer normalization (Ba et al., 2016). $\boldsymbol{W}_Q^{l,h}, \boldsymbol{W}_K^{l,h}, \boldsymbol{W}_V^{l,h} \in \mathbb{R}^{d\times\frac{d}{H}}$, $\boldsymbol{W}_O^l \in \mathbb{R}^{d\times d}$, $\boldsymbol{W}_1^l \in \mathbb{R}^{d\times 4d}$ and $\boldsymbol{W}_2^l \in \mathbb{R}^{4d\times d}$ are learnable projection weights. We hide the bias terms for simplification. $\boldsymbol{X}^0$ ($\boldsymbol{X}^l$ with $l=0$) is normally set as $\boldsymbol{E}^0$, and will be added with an extra learnable token $\boldsymbol{c}^0$ when performing classification tasks, *i.e.,* $\boldsymbol{X}^0 = [\boldsymbol{c}^0; \boldsymbol{E}^0]$. The feature is extracted from the same location of the last-layer sequence, which is $\boldsymbol{f} = \mathrm{LN}(\boldsymbol{c}^L)$.

The components of the Vision Transformer (ViT) are detailed in Table 9. The total number of parameters in ViT can be calculated as follows: $12Ld^2 + (13L + m + d_0 + 6)d$, where m denotes the number of separated image patches, d_0 represents the dimension of each image patch, and d represents the dimension of the embedding features. For example, in the case of ViT-B/16, where $m = \frac{224}{16} \times \frac{224}{16} = 196$, $d_0 = 16 \times 16 \times 3 = 768$, $d = 768$, $L = 12$, the parameter quantity amounts to 85,799,424 ($\approx$ 85.80M).

Table 9: Model architecture and parameter quantity for CLIP-ViT.

Layers	Components	Variables	Size	#Params.
Embedding	Projection	-	$d_0 \times d$	$(m + d_0 + 2)d$
	Class Token	$\boldsymbol{c}^0$	d	
	Positional	-	$(m+1) \times d$	
	LN	$\boldsymbol{\gamma}, \boldsymbol{\beta}$	d, d	$2d$
Block-1 ($l=1$)	LN	$\boldsymbol{\gamma}, \boldsymbol{\beta}$	d, d	$12d^2 + 13d$
	MSA	$\{\boldsymbol{W}_Q^{l,h}, \boldsymbol{b}_Q^{l,h}\}_{h=1}^H$ $\{\boldsymbol{W}_K^{l,h}, \boldsymbol{b}_K^{l,h}\}_{h=1}^H$ $\{\boldsymbol{W}_V^{l,h}, \boldsymbol{b}_V^{l,h}\}_{h=1}^H$ $\boldsymbol{W}_O^l, \boldsymbol{b}_O^l$	$\{d \times \frac{d}{H}, \frac{d}{H}\} \times H$ $\{d \times \frac{d}{H}, \frac{d}{H}\} \times H$ $\{d \times \frac{d}{H}, \frac{d}{H}\} \times H$ $d \times d, d$	
	LN	$\boldsymbol{\gamma}, \boldsymbol{\beta}$	d, d	
	FFN	$\boldsymbol{W}_1^l, \boldsymbol{b}_1^l$ $\boldsymbol{W}_2^l, \boldsymbol{b}_2^l$	$d \times 4d, 4d$ $4d \times d, d$	
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
Block-L	$\cdots$	$\cdots$	$\cdots$	$12d^2 + 13d$
	LN	$\boldsymbol{\gamma}, \boldsymbol{\beta}$	d, d	$2d$

Representative Structured Lightweight Fine-Tuning Methods. In this paper, we explore lightweight fine-tuning by learning a few learnable parameters for the adaptation. This approach prevents the decline of generalization ability benefiting from the foundation model. As only a small set of task-specific parameters is introduced, the model not only mitigates overfitting but also exhibits rapid convergence. Concretely, we propose a unified framework LIFT. This framework is versatile and inclusive, allowing for the incorporation of a range of structured modules, including but not limited to

- *Bias-terms Fine-tuning* (*BitFit*) (Zaken et al., 2022) aims to fine-tune only the bias parts of the model. Formally, given a projection function $\boldsymbol{X}\boldsymbol{W} + \boldsymbol{b}$, it freezes $\boldsymbol{W}$ and optimizes $\boldsymbol{b}$.
- *Visual Prompt Tuning* (*VPT*) (Jia et al., 2022) prepends learnable prompts $\boldsymbol{P}^l \in \mathbb{R}^{p\times d}$ at each layer to extend $\boldsymbol{X}^l = [\boldsymbol{c}^l; \boldsymbol{E}^l]$ to $[\boldsymbol{c}^l; \boldsymbol{P}^l; \boldsymbol{E}^l]$. It has two variations: 1) *VPT-Shallow*, which only prepends prompts at the first layer; 2) *VPT-Deep*, which prepends prompts at all layers.
- *Adapter* (Houlsby et al., 2019) proposes to optimize a bottleneck module. The definition is $\mathrm{Adapter}(\boldsymbol{X}) = \mathrm{ReLU}(\mathrm{LN}(\boldsymbol{X})\boldsymbol{W}_{\text{down}})\boldsymbol{W}_{\text{up}}$, where $\boldsymbol{W}_{\text{down}} \in \mathbb{R}^{d\times r}$ and $\boldsymbol{W}_{\text{up}} \in \mathbb{R}^{r\times d}$ ($r \ll d$). In practical, it can be appended to the FFN layer to reconstruct $\mathrm{FFN}(\cdot)$ to $\mathrm{Adapter}(\mathrm{FFN}(\cdot))$.
- *Low-Rank Adapter (LoRA)* (Hu et al., 2022) is applied to the weights in MSA module. Specifically, it optimizes $\boldsymbol{W}_{\text{down}}$ and $\boldsymbol{W}_{\text{up}}$ to update $\boldsymbol{W}$ (*e.g.,* $\boldsymbol{W}_Q$ or $\boldsymbol{W}_V$) to $\boldsymbol{W} + \boldsymbol{W}_{\text{down}}\boldsymbol{W}_{\text{up}}$.

- *AdaptFormer* (Chen et al., 2022) changes the sequential *Adapter* to a parallel one. Formally, it computes $s \cdot \text{Adapter}(\hat{\boldsymbol{X}}^l)$ and adds it to $\boldsymbol{X}^l$ in Equation (15). Here, s can be a manually set or learnable scaling parameter[3].

We present the parameter quantities of the structured lightweight fine-tuning modules in Table 10. For all modules, the parameter quantities are at the polynomial level of d. Notably, p for VPT and r for Adapter are much smaller than d. In comparison to the entire Transformer block, a lightweight module is significantly low-complexity ($\mathcal{O}(d)$ vs. $\mathcal{O}(d^2)$).

Moreover, the parameter quantity for a classifier is approximately Kd, where K is the number of classes. In LIFT, we set the bottleneck dimension $r = 2^{\lfloor \log_2 (\frac{K}{2L}) \rfloor} \leq \frac{K}{2L}$ for the AdaptFormer module, so that the total parameter quantity is $L \cdot 2rd \leq Kd$ (ignoring constant terms). As a result, it learns even fewer parameters than the classifier.

Table 10: Parameter quantities for structured lightweight fine-tuning modules in a Transformer block.

Modules	Components	Variables	Size	#Params.
BitFit	LN-bias	$\boldsymbol{\beta}$	d	$11d$
	MSA-bias	$\{\boldsymbol{b}_Q^{l,h}\}_{h=1}^H$	$\{\frac{d}{H}\} \times H$	
		$\{\boldsymbol{b}_K^{l,h}\}_{h=1}^H$	$\{\frac{d}{H}\} \times H$	
		$\{\boldsymbol{b}_V^{l,h}\}_{h=1}^H$	$\{\frac{d}{H}\} \times H$	
		$\boldsymbol{b}_O^l$	d	
	LN-bias	$\boldsymbol{\beta}$	d	
	FFN-bias	$\boldsymbol{b}_1^l$	$4d$	
		$\boldsymbol{b}_2^l$	d	
VPT	Prompts	$\boldsymbol{P}^l$	$p \times d$	pd
Adapter	LN	$\boldsymbol{\gamma}, \boldsymbol{\beta}$	d, d	$(2r+3)d + r$
	Projection	$\boldsymbol{W}_{\text{down}}, \boldsymbol{b}_{\text{down}}$	$d \times r, r$	
		$\boldsymbol{W}_{\text{up}}, \boldsymbol{b}_{\text{up}}$	$r \times d, d$	
LoRA	Projection	$\boldsymbol{W}_{\text{down}}, \boldsymbol{W}_{\text{up}}$ (for $\boldsymbol{W}_Q$)	$d \times r, r \times d$	$4rd$
		$\boldsymbol{W}_{\text{down}}, \boldsymbol{W}_{\text{up}}$ (for $\boldsymbol{W}_V$)	$d \times r, r \times d$	
AdaptFormer	LN	$\boldsymbol{\gamma}, \boldsymbol{\beta}$	d, d	$(2r+3)d + r + 1$
	Projection	$\boldsymbol{W}_{\text{down}}, \boldsymbol{b}_{\text{down}}$	$d \times r, r$	
		$\boldsymbol{W}_{\text{up}}, \boldsymbol{b}_{\text{up}}$	$r \times d, d$	
	Scaling	s	1	

Impact of the Quantity of Learnable Parameters. In LIFT, we can define the amounts of learnable parameters. In Figure 6, we study how much the parameters impact performance by controlling the bottleneck dimension r. Overall, we find that the performance is robust to the change of dimensions and it achieves the best results when employing comparable learnable parameters to the classifier.

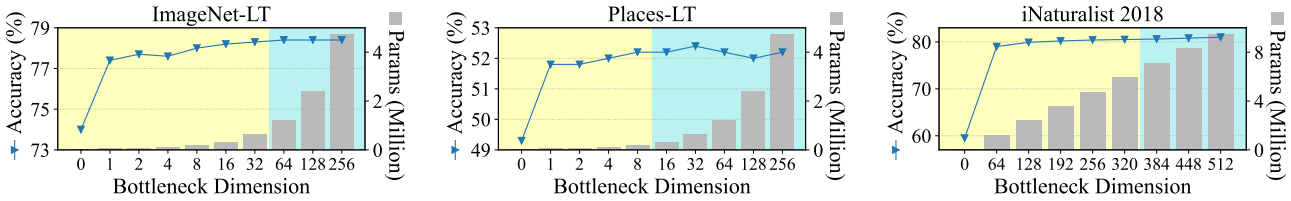


Figure 6: Comparison of different learnable parameters by changing the bottleneck dimension r. In the yellow area, the incorporated module has fewer learnable parameters than the classifier. The blue area is just the opposite.

[3] https://github.com/ShoufaChen/AdaptFormer/blob/main/models/adapter.py

Learned Representations of Varying Fine-Tuning Methods. In Figures 7 and 8, we illustrate the inter-class feature separabilities and intra-class distance distributions, based on the representation learned by (a) original CLIP, (b) full fine-tuning, (c) arbitrary lightweight fine-tuning, and (d-i) structured lightweight fine-tuning methods.

Compared to the original CLIP, all of these lightweight fine-tuning methods contribute to more distinctive representations. Among these methods, VPT-shallow may yield relatively weaker effects. This may be attributed to its design since VPT-shallow only prepends learnable prompts at the first layer. Beyond this, the other methods enable the feature separability even comparable to full fine-tuning. Notably, Adapter and AdaptFormer are more effective in enhancing separability, which is aligned with their superior performance in Table 6.

Compared to full fine-tuning, all of these lightweight fine-tuning methods yield undistorted intra-class distributions. For both head and tail classes, the features of training and test data almost overlap under the same distribution, which is similar to the original CLIP. This brings about their stable performance improvements, especially on tail classes, as shown in Table 6.

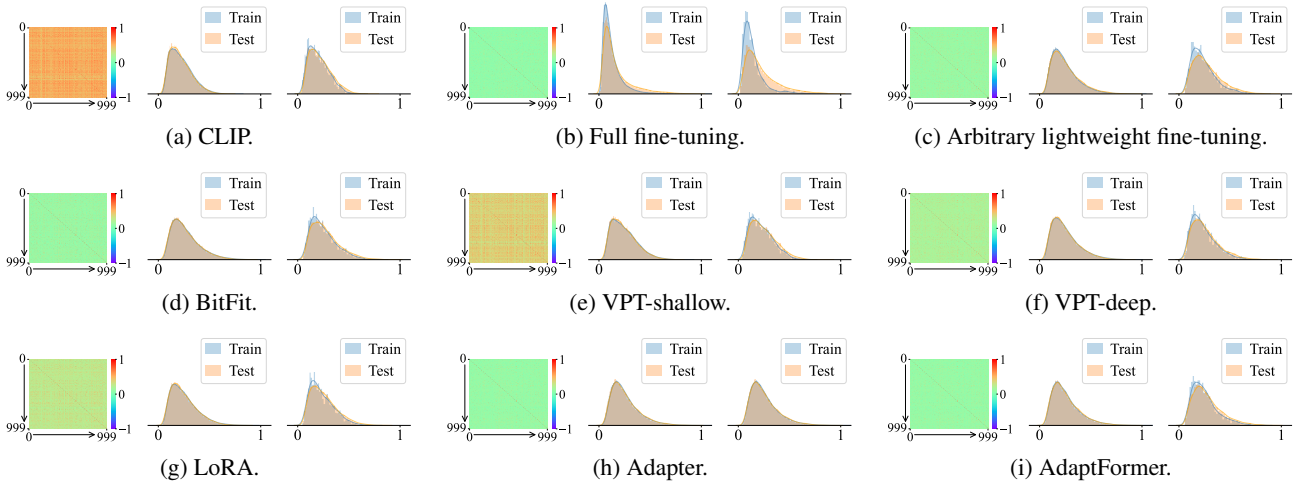


Figure 7: Visualization of the inter-class feature similarities (the heatmaps) and intra-class distance distributions from head classes (the left histograms) and tail classes (the right histograms) on ImageNet-LT.

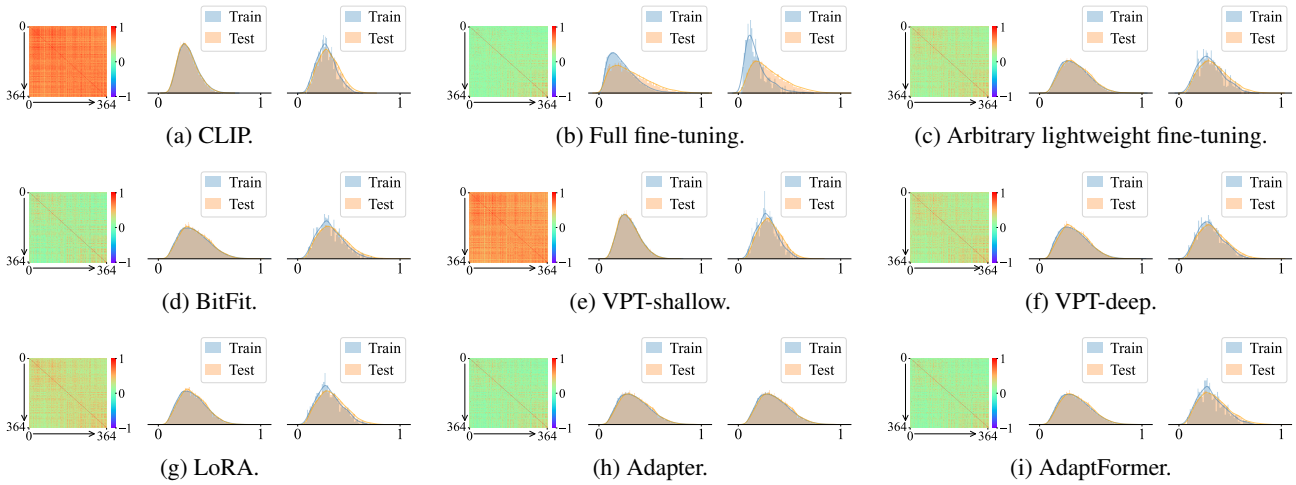


Figure 8: Visualization of the inter-class feature similarities (the heatmaps) and intra-class distance distributions from head classes (the left histograms) and tail classes (the right histograms) on Places-LT.

Effects of the Structured Lightweight Fine-Tuning Module on Each Layer. In each layer, the output of the AdaptFormer module is multiplied by a learnable scaling parameter s before being added to the corresponding block. Therefore, we can compare the values of s to analyze the effects of the module for different layers. In Figure 9, we visualize the value of

s from each layer. It is inspiring that AdaptFormer can adaptively learn suitable scaling parameters for different layers. Moreover, the values of the last layers tend to be larger, which indicates that the adaptation of the last several layers is more significant for downstream classification tasks.

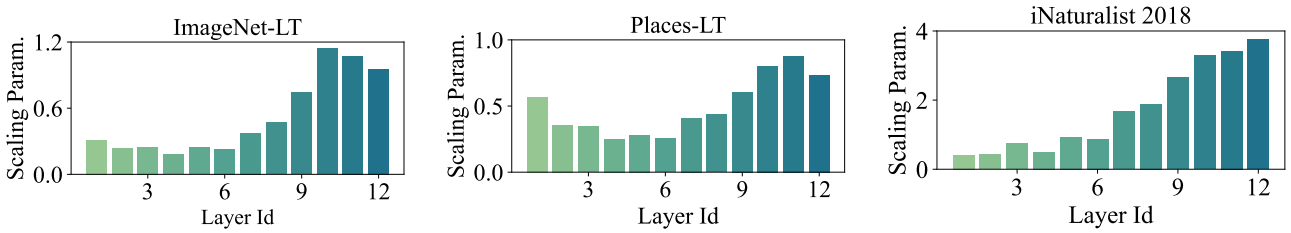


Figure 9: Learned scaling parameters of the AdaptFormer modules in different layers. AdaptFormer adaptively learns suitable scaling parameters for each layer, and the last several layers tend to have larger scaling parameters.

Lightweight Fine-Tuning vs. Partial Fine-Tuning. Partial fine-tuning (He et al., 2022) is an intuitive way to reduce the learnable parameters and avoid overfitting. Specifically, it fine-tunes the last k layers of Transformer blocks while freezing the others. In Figure 10, we compare partial fine-tuning and lightweight fine-tuning (LIFT) on ImageNet-LT, Places-LT, and iNaturalist 2018. Similar to full fine-tuning, partial fine-tuning is also sensitive to learning rate. When k is small (*e.g.,* $0, 1, 2$), a higher learning rate is better. However, when k is large (*e.g.,* $9, 12$), the high learning rate leads to a severe deterioration in the accuracy, whereby a smaller learning rate is more appropriate. Moreover, even if we have searched for the optimal learning rate, it is non-trivial to choose the number of fine-tuned layers k for different datasets, since the best k is 2 for ImageNet-LT, 1 for Places-LT, and 6 for iNaturalist 2018. In contrast, lightweight fine-tuning consistently performs well with fixed hyperparameters.

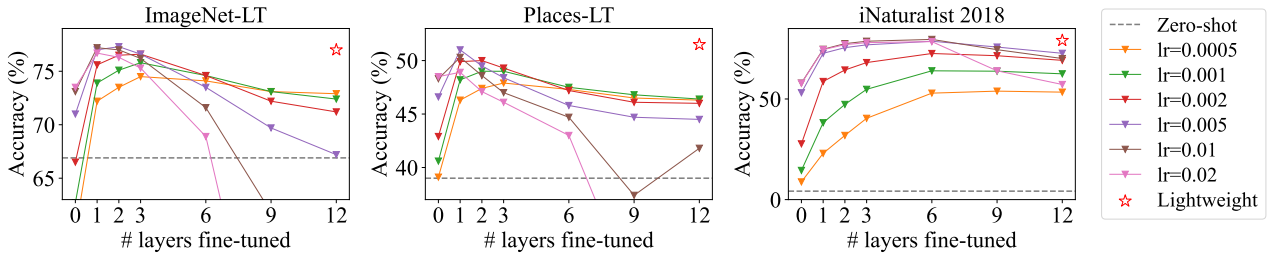


Figure 10: Partial fine-tuning the last k layers. Both methods use cosine classifier and semantic-aware initialization for fair comparison. Similar to full fine-tuning, we search learning rate from $\{0.02, 0.01, 0.005, 0.002, 0.001, 0.0005\}$ for partial fine-tuning. For LIFT, we fix the learning rate to 0.01. Partial fine-tuning needs to elaborately choose the proper learning rate and the fine-tuned layers for the best performance, while LIFT consistently performs optimally.

Time Cost Analysis. We also record the time cost of each structured lightweight fine-tuning method and report the results in Table 11. The results suggest that the time costs for different methods are highly close.

Table 11: Training time per epoch when using different structured lightweight fine-tuning methods.

Methods		**ImageNet-LT**	**Places-LT**
LIFT w/	BitFit	2 m 32 s	1 m 23 s
	VPT-shallow	2 m 31 s	1 m 22 s
	VPT-deep	2 m 40 s	1 m 27 s
	Adapter	2 m 37 s	1 m 25 s
	LoRA	2 m 33 s	1 m 23 s
	AdaptFormer	2 m 40 s	1 m 28 s

E. Explanation of Test-Time Ensembling

We present the detailed procedures of *test-time ensembling* (TTE) in Algorithm 1, using ViT-B/16 (224×224 resolution) as the backbone model. The highlighted lines denote the additional steps introduced by TTE. Conventionally, an image is first resized and center-cropped, and then split into patches before being fed into the Transformer model. However, this approach inevitably leads to the segmentation of important patterns across different patches, thus impeding the generalization. By employing diverse croppings, patterns that might be segmented in one cropping will be preserved in another. It is crucial to emphasize that the expanded size e should not be a multiple of the patch size 16; otherwise, the five cropped images will share a significant portion of the same patches, rendering the expected diversity unattainable. In LIFT, we default to set $e = 24$. Furthermore, we conduct a comparison of different expanded sizes and report the results in Figure 11. Aside from TTE, we explore other augmentation techniques such as TTE + Flipping or Random Augmentations (He et al., 2016) multiple times. The results in Table 12 demonstrate that TTE is more effective than other augmentation methods.

Algorithm 1 TEST-TIME ENSEMBLING

Input: Image $\boldsymbol{x}$, expanded size e, input resolution (224).

1: Resize $\boldsymbol{x}$ to $\boldsymbol{x}'$ sized $(224 + e) \times (224 + e)$.
2: Crop the center 224×224 portion of $\boldsymbol{x}'$, denoted by $\boldsymbol{x}^{\mathrm{c}}$.
3: Split $\boldsymbol{x}^{\mathrm{c}}$ evenly into m patches $[\boldsymbol{x}_1^{\mathrm{p}}; \cdots; \boldsymbol{x}_m^{\mathrm{p}}]$ (each $\boldsymbol{x}_i^{\mathrm{p}}$ is sized 16×16, and $m = \frac{224}{16} \times \frac{224}{16} = 196$).
4: Calculate the feature $\boldsymbol{f}^{\mathrm{c}}$ and then the logits $\boldsymbol{z}^{\mathrm{c}}$.
5: Crop the top left 224×224 portion of $\boldsymbol{x}'$, repeat procedure 3-5 and obtain the logits $\boldsymbol{z}^{\mathrm{tl}}$.
6: Crop the top right 224×224 portion of $\boldsymbol{x}'$, repeat procedure 3-5 and obtain the logits $\boldsymbol{z}^{\mathrm{tr}}$.
7: Crop the bottom left 224×224 portion of $\boldsymbol{x}'$, repeat procedure 3-5 and obtain the logits $\boldsymbol{z}^{\mathrm{bl}}$.
8: Crop the bottom right 224×224 of portion $\boldsymbol{x}'$, repeat procedure 3-5 and obtain the logits $\boldsymbol{z}^{\mathrm{br}}$.

Output: Predicted logits $\boldsymbol{z} = \mathrm{Average}(\boldsymbol{z}^{\mathrm{c}} + \boldsymbol{z}^{\mathrm{tl}} + \boldsymbol{z}^{\mathrm{tr}} + \boldsymbol{z}^{\mathrm{bl}} + \boldsymbol{z}^{\mathrm{br}})$.

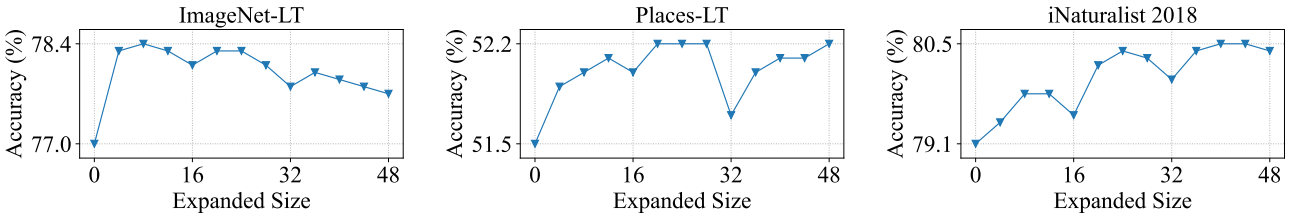


Figure 11: Performance of TTE with different expanded size e. Setting $e = 0$ indicates not applying TTE. Setting e to a multiple of the patch size 16 yields suboptimal performance. Generally, $e = 24$ is suitable for enhancing the generalization.

Table 12: Comparison of different augmentation methods on ImageNet-LT.

Augmentation methods	Augmentation times	Overall	Head	Medium	Tail
TTE (Ours)	5	**78.3**	**81.3**	**77.4**	**73.4**
TTE + Flipping	10	**78.3**	**81.3**	77.3	73.3
Random Augmentation	5	76.7	79.7	75.5	71.9
Random Augmentation	10	77.3	80.3	76.3	72.2
Random Augmentation	15	77.7	80.8	76.6	72.3
Random Augmentation	20	77.8	80.9	76.7	72.7

F. Textual Prompts for Semantic-Aware Initialization

In LIFT, we use "`a photo of a [CLASS].`" as the template to generate textual prompts and then compute their features to initialize the classifier weights. One may be concerned with the impact of the used prompts. We conduct experiments to compare different prompting methods, including 1) the original class name ("`[CLASS]`") and 2) prompt ensembling (Radford et al., 2021) which applies different templates to class names. The results in Table 13 show that these prompts have similar performances and that using "`a photo of a [CLASS].`" is adequate for generalization.

Moreover, we posit that CLIP has seen sufficient language corpus, considering its pre-training on web-scale datasets. However, it is noteworthy that CLIP may fail to recognize specific class names. This probably stems from its limited vocabulary size (CLIP contains approximately 49K vocabulary) or encountering uncommon or novel concepts. In this case, semantic-aware initialization may regress to random initialization. In response to this, we explore an alternative approach by incorporating class descriptions (which can be crafted manually or generated using large language models). In practice, we follow Menon & Vondrick (2023) to generate the descriptions for each class, then combine these descriptions and calculate the textual feature for initialization. The results are reported in the bottom line of Table 13, which shows that the use of class descriptions can also enhance the performance compared to random initialization.

Table 13: Comparison of different prompting methods on ImageNet-LT.

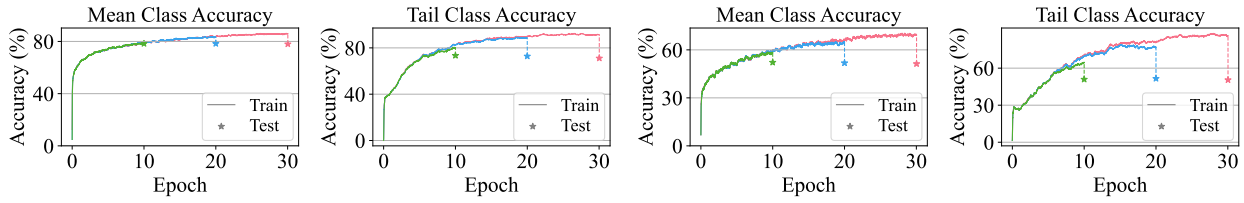
Prompting methods	Overall	Head	Medium	Tail
None prompt (random initialization)	76.1	80.8	75.9	63.2
"`[CLASS]`"	78.2	**81.4**	77.3	72.3
"`a photo of a [CLASS].`"	**78.3**	81.3	**77.4**	**73.4**
Prompt ensembling	**78.3**	81.3	**77.4**	73.3
Class descriptions (w/o class names)	77.4	81.3	76.9	68.2

G. Additional Experiments

Comparison of Different Training Epochs. In LIFT, we train 10 epochs on ImageNet-LT and Places-LT, and 20 epochs on iNaturalist 2018 considering its large data scale. In Table 14, we report the results of training different epochs. On ImageNet-LT and Places-LT, increasing the training epochs does not yield significant improvements. Generally, 10-20 epochs are appropriate for most cases. We also visualize the training and test accuracy as a function of epochs in Figure 12. When trained for more epochs (>20), LIFT converges with higher training accuracy. Nonetheless, the test accuracy shows no corresponding enhancement. On iNaturalist 2018, when training for 5 epochs, LIFT achieves an overall accuracy of 67.3% (w/o TTE) / 68.6% (w/ TTE), which surpasses Decoder (Wang et al., 2024) by more than 8% (please refer to Table 3 for comparison). Moreover, by training more epochs (*e.g.,* 30 epochs), LIFT achieves an additional performance improvement by 1%. However, this will increase the computational overhead, so we abort this approach in LIFT.

Table 14: Results of LIFT (with and without TTE) on iNaturalist 2018 by training different epochs.

Methods	#Epochs	ImageNet-LT				Places-LT				iNaturalist 2018			
		Overall	Head	Med.	Tail	Overall	Head	Med.	Tail	Overall	Head	Med.	Tail
LIFT (Ours)	5	76.2	79.9	75.8	67.2	50.7	51.2	52.0	46.6	67.3	70.4	71.0	61.8
	10	77.0	80.2	76.1	71.5	51.5	51.3	52.2	50.5	76.1	71.3	75.9	77.5
	20	77.1	80.7	75.9	71.4	51.1	51.0	51.3	51.0	79.1	72.4	79.0	81.1
	30	76.9	81.1	75.6	69.4	50.3	50.9	50.1	49.5	80.1	73.8	80.0	81.9
LIFT w/ TTE (Ours)	5	77.5	80.9	77.2	69.0	51.3	**51.7**	53.0	46.8	68.6	70.5	72.3	63.5
	10	**78.3**	81.3	**77.4**	**73.4**	**52.2**	**51.7**	**53.1**	50.9	77.3	71.9	77.1	78.9
	20	**78.3**	81.8	77.1	72.8	51.8	51.3	52.3	**51.6**	80.4	74.0	80.3	82.2
	30	78.0	**82.2**	76.6	71.1	51.3	51.4	51.5	50.5	**81.3**	**75.1**	**81.2**	**83.0**



(a) ImageNet-LT. (b) Places-LT.

Figure 12: Convergence curve of mean class and tail class accuracy.

Comparison of Different Classifiers. In LIFT, we default to optimize the cosine classifier with a scaling factor $\sigma = 25$. We propose to use the cosine classifier to overcome the biased weight norms. We further assess the linear classifier $z_k = \boldsymbol{w}_k^\top \boldsymbol{f} + b$, the L2-normalized classifier $z_k = \frac{\boldsymbol{w}_k^\top}{\|\boldsymbol{w}_k\|_2}\boldsymbol{f}$, as well as the cosine classifier with $\sigma \in \{15, 20, 25, 30, 35\}$, and report the results in Table 15. The results show that the linear classifier performs well on ImageNet-LT and Places-LT, but unsatisfactorily on the more challenged iNaturalist 2018 dataset. This can be inferred from the classifier weight norms shown in Figure 13, where the weight norms of iNaturalist 2018 are much more skewed. By removing the impact of weight norms, the L2-normalized classifier achieves higher performance, especially on the tail classes. When adopting the cosine classifier, setting σ to 25 or 30 leads to the best performance. Without loss of generality, we default to set $\sigma = 25$.

Table 15: Performance of LIFT with different classifiers.

Classifiers		ImageNet-LT				Places-LT				iNaturalist 2018			
		Overall	Head	Med.	Tail	Overall	Head	Med.	Tail	Overall	Head	Med.	Tail
Linear		78.2	81.2	77.2	72.8	**52.3**	51.7	52.8	52.0	75.7	**75.8**	77.4	73.6
L2-normalized		78.4	81.2	77.2	**74.7**	52.2	51.3	52.7	**52.4**	80.0	74.1	79.9	81.8
Cosine	$\sigma = 15$	75.3	81.1	76.1	55.9	49.4	**52.8**	52.8	35.3	76.5	73.4	76.4	77.4
	$\sigma = 20$	77.5	81.1	77.1	69.1	51.6	52.2	**53.3**	46.8	79.6	73.9	79.3	81.6
	$\sigma = 25$	78.3	81.3	**77.4**	73.4	52.2	51.7	53.1	50.9	**80.4**	74.0	80.3	**82.2**
	$\sigma = 30$	**78.5**	**81.5**	77.3	74.2	52.1	51.5	52.6	51.8	80.3	73.8	**80.4**	81.9
	$\sigma = 35$	78.4	**81.5**	77.1	73.9	51.7	51.3	52.1	51.7	79.8	74.1	79.9	81.3

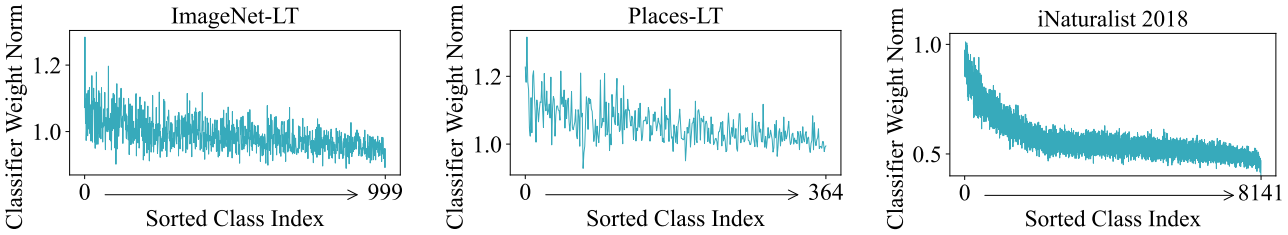


Figure 13: Weight norms of the learned linear classifier on three long-tail datasets. Classes are sorted by their frequency in the training dataset. On iNaturalist 2018, the weight norms are much more imbalanced, leading to a suboptimal performance of the linear classifier.

Comparison of Different Losses. In Table 16, we compare the performance of LIFT with different losses, including cross-entropy (CE) loss, focal loss (Lin et al., 2017), label-distribution-aware margin (LDAM) loss (Cao et al., 2019), class-balanced (CB) loss (Cui et al., 2019), generalized re-weighting (GRW) loss (Zhang et al., 2021), label distribution disentangling (LADE) loss (Hong et al., 2021). The results are shown in Table 16, wherein the LA loss achieves the highest performance among all of the cases. In contrast, the other losses such as LDAM and LADE can not achieve satisfactory performance in all cases. Moreover, we give a theoretical proof of the LA loss and analyze the impact of the class-conditional distribution in Appendix B,.

Table 16: Performance of LIFT with different losses.

Losses	ImageNet-LT				Places-LT				iNaturalist 2018			
	Overall	Head	Med.	Tail	Overall	Head	Med.	Tail	Overall	Head	Med.	Tail
CE	71.8	86.1	68.7	42.1	42.1	56.7	38.0	24.5	74.8	82.1	75.4	72.0
Focal (Lin et al., 2017)	72.1	85.5	69.1	44.7	42.7	56.1	38.8	26.8	73.0	81.1	74.0	69.6
LDAM (Cao et al., 2019)	69.6	**86.4**	66.5	33.4	40.4	**56.8**	36.0	20.1	75.9	**84.3**	77.0	72.4
CB (Cui et al., 2019)	76.9	82.3	76.3	63.5	50.0	52.6	51.5	41.9	78.6	71.6	79.0	79.8
GRW (Zhang et al., 2021)	76.9	82.3	76.3	63.7	50.1	52.4	51.7	42.0	78.6	71.9	79.1	79.8
LADE (Hong et al., 2021)	78.0	81.2	76.7	**73.4**	51.2	51.3	51.7	49.6	**80.4**	73.8	80.0	**82.5**
LA (Menon et al., 2021)	**78.3**	81.3	**77.4**	**73.4**	**52.2**	51.7	**53.1**	**50.9**	**80.4**	74.0	**80.3**	82.2

More Detailed Observations on Model Convergence. In Figures 14 and 15, we illustrate the convergence curve on training loss and accuracy. We report the mean class accuracy, as well as the head, medium, and tail class accuracy. The results show that LIFT converges rapidly with 10 training epochs. Without the structured lightweight fine-tuning (SLF) module, the training loss and accuracy converge suboptimally on all classes. Without semantic-aware initialization (SAI), the head-class accuracy is slightly affected, while the tail-class accuracy decreases by a large margin.

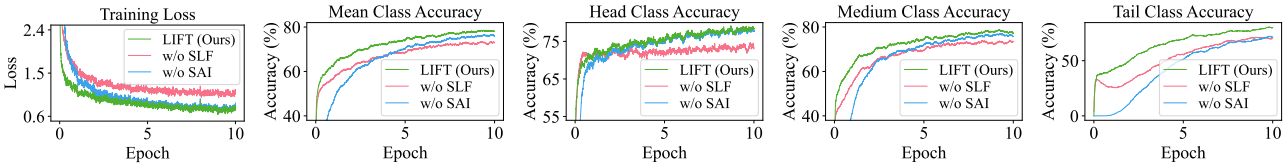


Figure 14: Convergence curves of training loss and accuracy on ImageNet-LT.

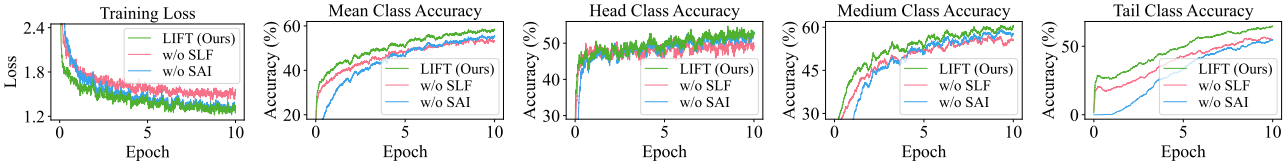


Figure 15: Convergence curves of training loss and accuracy on Places-LT.

LIFT with Variant Backbones. In addition to ViT-B/16, we also assess LIFT based on the larger ViT-L/14. The results in Tables 17 to 20 show that LIFT surpasses the state-of-the-art method Decoder (Wang et al., 2024) by a considerable margin, showcasing improvements of 3.1% on ImageNet-LT, 5.0% on Places-LT and 12.1% on iNaturalist 2018. Additionally, the model with higher resolution (336×336 pixels) yields improved performance. Furthermore, the incorporation of TTE consistently enhances the generalization. These results underscore the adaptability of LIFT across variant backbones.

Table 17: Results on ImageNet-LT with ViT-L/14 as the backbone.

Methods	**Backbone**	**Learnable Params.**	**#Epochs**	**Overall**	**Head**	**Medium**	**Tail**
Zero-shot CLIP	ViT-L/14	-	-	73.6	74.6	73.1	72.3
Decoder (Wang et al., 2024)	ViT-L/14	39.79M	∼18	79.3	-	-	-
LIFT (Ours)	ViT-L/14	**0.86M**	**10**	82.4	84.8	81.7	78.0
LIFT w/ TTE (Ours)	ViT-L/14	**0.86M**	**10**	**82.9**	**85.3**	**82.2**	**78.6**

Table 18: Results on Places-LT with ViT-L/14 as the backbone.

Methods	**Backbone**	**Learnable Params.**	**#Epochs**	**Overall**	**Head**	**Medium**	**Tail**
Zero-shot CLIP	ViT-L/14	-	-	39.9	38.1	39.2	45.1
Decoder (Wang et al., 2024)	ViT-L/14	39.79M	∼34	48.4	-	-	-
LIFT (Ours)	ViT-L/14	**0.27M**	**10**	53.4	52.6	**54.1**	53.3
LIFT w/ TTE (Ours)	ViT-L/14	**0.27M**	**10**	**53.7**	**53.1**	**54.1**	**53.8**

Table 19: Results on iNaturalist 2018 with ViT-L/14 as the backbone.

Methods	**Backbone**	**Learnable Params.**	**#Epochs**	**Overall**	**Head**	**Medium**	**Tail**
Zero-shot CLIP	ViT-L/14	-	-	5.8	11.1	5.4	4.9
Decoder (Wang et al., 2024)	ViT-L/14	39.79M	∼**5**	72.3	65.5	73.2	73.0
LIFT (Ours)	ViT-L/14	**6.37M**	20	84.4	79.3	84.6	85.5
LIFT w/ TTE (Ours)	ViT-L/14	**6.37M**	20	**85.2**	**80.2**	**85.1**	**86.6**

Table 20: Results on iNaturalist 2018 with ViT-L/14 (336×336 pixels) as the backbone.

Methods	Backbone	Learnable Params.	#Epochs	Overall	Head	Medium	Tail
Zero-shot CLIP	ViT-L/14@336px	-	-	6.2	11.5	5.8	5.3
LIFT (Ours)	ViT-L/14@336px	**6.37M**	20	87.0	83.2	87.0	87.9
LIFT w/ TTE (Ours)	ViT-L/14@336px	**6.37M**	20	**87.4**	**83.6**	**87.4**	**88.3**

LIFT employs ViT as its backbone, and there may be concerns regarding the adoption of the widely used ResNet (He et al., 2016). However, due to the absence of a dedicated structured lightweight fine-tuning method tailored for ResNet, it is challenging to integrate our method with ResNet. Despite this limitation, we have explored some straightforward strategies, including 1) incorporating a scaling and shifting (SSF) (Lian et al., 2022) module after the backbone, and 2) fine-tuning solely the bias terms of ResNet. The results are presented in Tables 21 and 22. In comparison to zero-shot CLIP and previous methods reported in Tables 1 and 2, our method achieves significantly superior performance with lower computational costs.

Table 21: Results on ImageNet-LT with ResNet-50 as the backbone. All methods use TTE for fair comparison.

Methods	Backbone	Learnable Params.	#Epochs	Overall	Head	Medium	Tail
Zero-shot CLIP	ResNet-50	-	-	57.6	58.6	56.9	56.9
LIFT w/ SSF	ResNet-50	0.002M	10	66.9	72.0	66.5	54.1
LIFT w/ bias tuning	ResNet-50	0.034M	10	67.8	72.0	67.3	57.6
LIFT w/ bias tuning & SSF	ResNet-50	0.036M	10	**68.3**	**72.5**	**67.8**	**58.2**

Table 22: Results on Places-LT with ResNet-50 as the backbone. All methods use TTE for fair comparison.

Methods	Backbone	Learnable Params.	#Epochs	Overall	Head	Medium	Tail
Zero-shot CLIP	ResNet-50	-	-	35.2	33.1	34.6	40.4
LIFT w/ SSF	ResNet-50	0.002M	10	46.7	47.5	48.7	40.7
LIFT w/ bias tuning	ResNet-50	0.034M	10	47.9	**48.2**	49.8	42.9
LIFT w/ bias tuning & SSF	ResNet-50	0.036M	10	**48.1**	48.1	**50.0**	**43.9**

Apart from the vision-language model CLIP, we have also validated LIFT using the ImageNet-21K pre-training model, which is a vision-only model. We employ the class mean features to initialize the classifier due to the lack of a corresponding text encoder. The results are provided in Tables 23 to 25. It is worth noting that the superior performance on ImageNet-LT may attributed to potential data leakage from ImageNet-21K. The performance on Places-LT is lower than CLIP pre-training, while still outperforming most state-of-the-art methods in Table 2. The performance on iNaturalist 2018 exceeds that of CLIP pre-training and outperforms all state-of-the-art methods in Table 3.

Table 23: Results of LIFT on ImageNet-LT with different pre-training models. All methods use TTE for fair comparison.

	Backbone	Learnable Params.	#Epochs	Overall	Head	Medium	Tail
CLIP Pre-training	ViT-B/16	0.62M	10	78.3	81.3	77.4	73.4
ImageNet-21K Pre-training	ViT-B/16	0.62M	10	**84.2**	**86.0**	**83.6**	**80.6**

Table 24: Results of LIFT on Places-LT with different pre-training models. All methods use TTE for fair comparison.

	Backbone	Learnable Params.	#Epochs	Overall	Head	Medium	Tail
CLIP Pre-training	ViT-B/16	0.18M	10	**52.2**	**51.7**	**53.1**	**50.9**
ImageNet-21K Pre-training	ViT-B/16	0.18M	10	49.2	48.9	50.2	47.4

Table 25: Results of LIFT on iNaturalist 2018 with different pre-training models. All methods use TTE for fair comparison.

	Backbone	Learnable Params.	#Epochs	Overall	Head	Medium	Tail
CLIP Pre-training	ViT-B/16	4.75M	20	80.4	74.0	80.3	82.2
ImageNet-21K Pre-training	ViT-B/16	4.75M	20	**81.9**	**74.9**	**82.3**	**83.3**

H. Related Work

Long-Tail Learning via Deep Learning. Conventional methods train convolutional neural network models like ResNet and ResNeXt on long-tail datasets. Concerning the class imbalance, there are three main directions to improve the performance: 1) data manipulation (Zhou et al., 2020; Kang et al., 2020; Yang & Xu, 2020; He et al., 2021; Park et al., 2022; Ahn et al., 2023; Shi et al., 2023; Gao et al., 2023), 2) representation learning (Liu et al., 2019; Kang et al., 2021; Wang et al., 2021b; Cui et al., 2021; Samuel & Chechik, 2021; Zhu et al., 2022; Yang et al., 2022; Peifeng et al., 2023; Ma et al., 2023), and 3) model output adjustment (Cao et al., 2019; Ren et al., 2020; Menon et al., 2021; Hong et al., 2021; Zhang et al., 2021; Wei et al., 2022; Han, 2023; Shi et al., 2024). Data manipulation typically includes designing re-sampling strategies, and data augmentations. Many works improve the performance by adopting two-stage training where the first stage learns representations and the second stage learns the classifier (Zhong et al., 2021; Wei & Gan, 2023; Nam et al., 2023). The adjustment of the model's outputs can be done during training by optimizing unbiased loss functions or after training. In contrast to the aforementioned works, this paper presents an end-to-end training framework that combines the advantages of foundation models and multiple existing techniques. We conduct in-depth research on how to properly utilize the foundation models and enable the unbiased loss function to achieve optimal effects.

Long-Tail Learning via Foundation Model. Fine-tuning foundation models such as CLIP (Radford et al., 2021) and ViT (Dosovitskiy et al., 2021) has attracted widespread attention (Steiner et al., 2021; Zhou et al., 2022b;a; Yu et al., 2023; Jia et al., 2024; Zhou et al., 2024), and has emerged as an effective strategy to address class imbalance due to the strong representation learning capabilities (Ma et al., 2021; Long et al., 2022; Tian et al., 2022; Iscen et al., 2023; Dong et al., 2023; Xia et al., 2023; He et al., 2023; Song et al., 2023; Wang et al., 2024; Li et al., 2024). However, it is important to note that these methods often require prolonged training time and, in some cases, rely on external training data to facilitate the learning process. In contrast, our proposed approach exhibits the remarkable ability to achieve convergence in fewer than 20 epochs and does not need external data. Furthermore, our method is general, allowing for seamless integration with various lightweight fine-tuning approaches.

I. Limitations

Limitations of Arbitrary Lightweight Fine-Tuning. Despite the remarkable performance achieved through arbitrary lightweight fine-tuning, this approach exhibits a slightly slower training process (3'10" on ImageNet-LT and 1'43" on Places-LT per epoch). When compared to structured lightweight fine-tuning methods in Table 11, its time cost is higher (approximately 1.2×). This is because the GPU has challenges in accelerating computation with unstructured parameters. Nonetheless, the proposed arbitrary lightweight fine-tuning is capable of achieving rapid convergence within 20 epochs, which is much more efficient compared to prior works. We will focus on enhancing the efficiency in our future work. In this paper, we propose arbitrary lightweight fine-tuning primarily to demonstrate the effectiveness of lightweight fine-tuning, as even arbitrarily selected parameters without guidance can still achieve superior performance.

Limitations of Semantic-Aware Initialization. In this paper, we propose semantic-aware initialization to leverage the semantic knowledge from CLIP and enhance the initialization. However, when deploying vision-only foundation models, integrating semantic knowledge becomes a challenging task. In such scenarios, we have identified that utilizing class mean features proves to be a viable choice. As delineated in Table 7, this approach leads to remarkable performance improvements, surpassing most of the existing methods in Tables 1 and 2. Additionally, Tables 4 and 23 to 25 also demonstrate the effectiveness of class mean features when adopted to ImageNet-21K pre-training vision models. It remains an intriguing challenge how to initialize the classifier for visual-only foundation models with long-tail data. Nonetheless, our findings demonstrate that opting for class mean features is an effective approach.