Meta-Learning Hyperparameters for Parameter Efficient Fine-Tuning

Supplementary Material

This supplementary material provides additional theoretical foundations, extended analyses, and experimental results that complement our main manuscript (references to the main manuscript are shown in red):

A. Theoretical Foundations

A.1. Equivalence and Advantages of Optimizing Scaling Factors rather than Learning Rate

This section motivates our choice of optimizing scaling factors instead of learning rates (supplementary to Section 3.1, line 298). We prove that optimizing scaling factors 1) is equivalent to optimizing the learning rate while 2) offers more flexible position-wise control over PEFT modules.

The fact that "learning any of them is equivalent" can be derived from LoRA's mechanism. We start with LoRA's formulation in the forward pass. The output y of the layer when given input x is:

$$y = W'x = (W + \alpha(AB))x, \tag{S1}$$

where W' denotes the updated weight matrix, W is the original pre-trained weight matrix, α is the scaling factor, and A, B are the low-rank decomposition matrices. Through gradient calculation during backpropagation (given loss L), we obtain the gradients of parameters A and B:

$$\frac{\partial L}{\partial A} = \alpha \cdot \frac{\partial L}{\partial y} x^\top \cdot B^\top, \quad \frac{\partial L}{\partial B} = \alpha \cdot A^\top \cdot \frac{\partial L}{\partial y} x^\top, \tag{S2}$$

where $\frac{\partial L}{\partial y}$ represents the gradient of the loss with respect to the layer's output y. During optimization, given learning rate η, the parameter updates follow:

$$A_{\text{new}} = A - \eta\alpha \cdot \frac{\partial L}{\partial y} x^\top \cdot B^\top, \quad B_{\text{new}} = B - \eta\alpha \cdot A^\top \cdot \frac{\partial L}{\partial y} x^\top. \tag{S3}$$

These update equations reveal a key insight: the magnitude of parameter updates is controlled by the product $\eta \cdot \alpha$. This indicates that optimizing scaling factors is equivalent to optimizing the learning rate. For example, doubling the scaling factor α has the same effect as doubling the learning rate η.

A.2. NP-hardness of MINLP

We show that the MINLP problem defined in Section 3.2, Formulation (6) is NP-hard through a polynomial-time reduction from the 0-1 knapsack problem, a well-known NP-hard problem.

0-1 Knapsack Problem. Given a set of n items, each with value $v_i > 0$ and weight $w_i > 0$, and a capacity $W > 0$, find a subset of items maximizing total value while keeping total weight within capacity:

$$\begin{aligned} \max_{\{x_i\}} \quad & \sum_{i=1}^{n} v_i x_i \\ \text{s.t.} \quad & \sum_{i=1}^{n} w_i x_i \leq W, \quad x_i \in \{0,1\}, \quad i = 1, \ldots, n \end{aligned} \tag{S4}$$

Reduction. Given any instance of the 0-1 knapsack problem, we construct an instance of the MINLP problem as follows:
1) Fix the position s to any value $s_0 \in \mathcal{S}$ and scaling factor α to any positive value α_0. These values remain constant throughout the reduction.
2) For each item i in the knapsack problem, create a corresponding layer depth $d_i = i$ where a PEFT module can be inserted. The binary decision x_i of selecting item i maps to whether a PEFT module is inserted at layer d_i.
3) Construct the validation loss function as $\mathcal{L}_{\text{val}} = -\sum_{i=1}^{n} v_i x_i$.
4) Map the knapsack constraint to the resource constraint in MINLP as $\sum_{i=1}^{n} w_i x_i \leq W$, where W represents the computational budget (*e.g.*, maximum number of PEFT modules allowed or FLOPs constraints) in our problem, and then the Formulation (6) becomes:

$$\begin{aligned} \min_{\{x_i\}} \quad & -\sum_{i=1}^{n} v_i x_i \\ \text{s.t.} \quad & \sum_{i=1}^{n} w_i x_i \leq W, \quad x_i \in \{0,1\}, \quad i = 1, \ldots, n \end{aligned} \tag{S5}$$

Correctness. The reduction is polynomial-time as it requires $O(n)$ operations. The solutions between the two problems have a one-to-one correspondence: a solution is feasible in the knapsack problem if and only if it is feasible in the MINLP problem, and the optimal solution to the MINLP problem corresponds to the optimal solution of the knapsack problem (differing only in sign). Since the 0-1 knapsack problem is NP-hard, this reduction proves that Formulation (6) in the main manuscript is also NP-hard.

A.3. Dicussion: Non-convexity of PEFT Optimization Objective.

Theoretically, our optimization objective is non-convex as it is on deep neural networks (DNN) [1]. Our optimization of plug-in modules (*e.g.*, LoRA) on DNN is also non-convex. *Empirically*, extensive prior works have shown that bi-level optimization is effective for non-convex problems of DNN [3, 4], and our results also show consistent and stable model convergence. In addition, we introduce two techniques to mitigate potential instabilities: 1) a softplus activation to constrain γ to be non-negative, and 2) dynamic sampling, which randomly selects 20% of the training data in meta-training loops to prevent overfitting to local optima.

B. Method Design and Analysis

B.1. Analysis on Data Sampling Strategies in Outer Loop

In Section 4.3, we demonstrate the effectiveness of random sampling in the outer loop of our bi-level optimization framework. Here, we further explore three alternative sampling strategies for the outer loop: class-balanced sampling, tail-heavy sampling, and larger sampling ratio. Results are provided in Table S1.

Class-Balanced Sampling. In Table 6, the random sampling strategy preserves the original long-tailed distribution. Here, we explore a balanced sampling strategy where each class contributes an equal number of samples. Compared to random sampling, class-balanced sampling achieves comparable overall accuracy (93.8% vs 93.6%) but slightly lower head class performance (-0.2%).

Tail-Heavy Sampling. We investigate a tail-heavy sampling strategy where tail classes are sampled with higher probability than head classes (inverse to their original frequencies). This approach achieves the highest tail-class accuracy (94.8%) among all sampling strategies, but at the cost of largely reduced head-class performance (-0.6%). The trade-off indicates that

Strategy	Head	Med	Tail	Avg
Random (20%)	93.2	94.6	93.1	93.6
Class-Balanced	93.0 ↓0.2	94.5 ↓0.1	93.3 ↑0.2	93.6 -
Tail-Heavy	92.0 ↓1.2	93.4 ↓1.2	94.8 ↑1.7	93.4 ↓0.2
Random (30%)	**93.5** ↑0.3	**94.8** ↑0.2	**93.9** ↑0.8	**94.1** ↑0.5

Table S1. **Accuracy (%) comparison of different sampling strategies.** We compare different sampling strategies in the outer loop of `MetaPEFT`. While tail-heavy sampling achieves the highest tail-class accuracy, it comes at the cost of reduced head-class performance. Random sampling with 30% ratio provides the best overall performance while maintaining balanced improvements across all classes. The arrows indicate changes compared to the 20% random sampling baseline (green for decrease, red for increase).

while emphasizing tail classes during meta-optimization can improve their representation, maintaining some balance with head classes remains important for overall model performance.

Sampling 30% of Training Data. Our observation in Table 6 shows that 30% sampling ratio achieves superior performance, we conduct comparison experiments using this setting. Results consistently show 0.5% improvement in average accuracy compared to 20% sampling, with particularly strong gains on tail classes ($+0.8\%$). However, this comes with a 15% increase in meta-optimization time.

B.2. More Implementation Details

We provide more implementation details as supplementary to Section 4.2.

B.2.1. Fair Comparisons Between Low-Rank and Adapter-based Methods

We compare the architecture difference between LoRA and additive methods, and show how we ensure a fair comparison between them.

Dataset	Dim	LoRA		Adapter		AdaptFormer	
		Tuner (M)	Head (M)	Tuner (M)	Head (M)	Tuner (M)	Head (M)
CIFAR100-IR100	4	0.249	0.076	0.101	0.076	0.101	0.076
FUSRS	4	0.197	0.005	0.136	0.005	0.136	0.005
DOTA	4	0.249	0.011	0.101	0.011	0.101	0.011
Places-LT	8	0.470	0.280	0.175	0.280	0.175	0.280
iNat2018	256	9.437	6.253	4.749	6.253	4.749	6.253

Table S2. **Comparison of learnable parameters across different PEFT methods and datasets.** "Dim" refers to the model dimension (*i.e.*, rank for LoRA, hidden dimension for Adapter/AdaptFormer). For each method, "Tuner (M)" refers to the number of learnable parameters within the PEFT modules, while "Head (M)" refers to those in the classifier head.

Module Size. We ensure a similar module size between LoRA and adapters. Concretely, we fix the rank of LoRA and the adapter's hidden dimension to a fixed and comparable size. We follow the setting in [5] to set higher hidden dimensions for larger classifier heads. The learnable parameters in different methods on different datasets are provided in Table S2.

B.2.2. Classifier Head Design and Initialization

We use a linear probing classifier head, which consists of a single linear layer initialized with class-mean features [6]. Specifically, for each class, we compute the mean of its feature representations from the training set (based on the frozen foundation model) and set the corresponding weights of the classifier to these mean vectors. This initialization leverages the inherent structure of the data, providing a robust starting point that enhances model convergence and performance.

B.2.3. Evaluation Protocols

We adopt the same evaluation protocols as in [5]. We report the macro-averaged per-class accuracies for head/medium/tail classes, and report the overall accuracy as the average of these three metrics. Especially, we use a test-time ensemble (TTE) strategy by averaging the predictions of three independent runs. While no dedicated literature focuses on TTE, this technique (also named test-time augmentation) has become a de facto standard in many implementations [5, 7].

C. Validation Experiments and Observations

C.1. Generalizing Key Observations to Natural Vision Domain

Position	CIFAR100			iNat2018			Places-LT			Avg_{tail}	Avg
	Head	Med	Tail	Head	Med	Tail	Head	Med	Tail		
K	87.3	84.6	84.0	63.2	72.3	73.3	46.5	47.0	46.4	67.9	67.9
Q	87.4	84.3	84.1	63.3	72.7	73.7	46.7	47.2	46.6	68.1	68.1
V	91.8	88.1	85.6	65.9	74.8	75.8	47.9	47.7	45.7	69.0	70.1
Out	91.8	88.2	85.4	65.8	75.3	75.3	47.5	47.8	46.0	68.9	70.1
MLP1	92.6	87.9	85.6	66.5	75.6	76.1	48.3	47.9	45.9	69.2	70.5
MLP2	92.3	88.1	86.6	65.9	75.1	76.1	47.9	47.9	45.7	69.5	70.4

Table S3. **Accuracy (%) comparison of intra-block positions in natural vision domain.** Results are reported on three transfer scenarios IN21K→{CIFAR100, iNaturalist-2018, and Places-LT}. "Out" denotes the attention output module, and "MLP1" and "MLP2" denote the first and second FFN linear layers. MLP1 achieves the highest overall accuracy (70.5%), while both MLP positions show strong tail-class performance (69.2-69.5%).

In the main manuscript, our major observations (Figure 1c and 1d) are based on the RS transfer scenario (*i.e.*, IN21K→DOTA). In this section, we extend our observations in the natural vision domain.

In Table S3, we conduct an ablation study on three standard long-tailed benchmarks in the natural vision domain. We can observe that: 1) MLP positions (MLP1 and MLP2) consistently outperform attention-based positions across all datasets, achieving the highest average accuracy (70.5% and 70.4%, respectively). 2) The performance gap is particularly pronounced in tail classes, where MLP positions achieve up to 76.1% accuracy on iNat2018, higher than attention positions (73.3-75.8%). These findings align with our observations in the RS domain, confirming that FFN layers are more effective insertion positions for PEFT modules.

C.2. Analysis of Position Combination Effects

Intra-Attn Positions				CIFAR100			iNat2018			Places-LT			Avg_{tail}	Avg
Q	V	K	Out	Head	Med	Tail	Head	Med	Tail	Head	Med	Tail		
✓	✓			91.2	87.9	86.8	66.5	75.4	76.9	48.4	47.6	45.4	69.70	69.57
✓	✓	✓		91.6	87.7	86.3	66.0	75.7	76.7	48.3	47.7	46.3	69.77	69.59
✓	✓	✓	✓	92.2	87.7	86.3	66.7	75.6	76.9	48.6	47.9	45.6	69.60	69.72

Table S4. **Accuracy (%) comparison of intra-attention positions' combinations.**

Intra-FFN Positions		CIFAR100			iNat2018			Places-LT			Avg_{tail}	Avg
MLP1	MLP2	Head	Med	Tail	Head	Med	Tail	Head	Med	Tail		
✓		92.6	87.9	85.6	66.5	75.6	76.1	48.3	47.9	45.9	69.20	69.60
	✓	92.3	88.1	86.6	65.9	75.1	76.1	47.9	47.9	45.7	69.47	69.51
✓	✓	92.4	87.9	86.8	67.9	76.3	76.7	48.4	48.3	45.7	69.73	70.04

Table S5. **Accuracy (%) comparison of intra-FFN positions' combinations.**

Beyond the individual position ablations presented in Table 2, we analyze how different positions interact when combined. We use two types of combinations: 1) combining different positions within attention blocks (Table S4), and 2) combining different positions within FFN layers (Table S5). From the results, we can observe: 1) For attention blocks, adding more positions does not necessarily lead to better performance. Concretely, using only Q and V achieves comparable or even slightly better tail performance than all positions (69.70% vs. 69.60% in Table S4 rows 1 and 3). 2) For FFN layers,

combining both MLP1 and MLP2 positions shows clear advantages in Table S5. The combined approach achieves the best overall performance (70.04%) and tail performance (69.73%), outperforming single position variants (69.60% and 69.51% for MLP1 and MLP2, respectively). 3) The improvements from position combinations are more pronounced in FFN layers compared to attention blocks. Concretely, FFN combinations show a $+0.53\%$ improvement in average accuracy (from 69.51% to 70.04% in Table S5), while attention combinations only yield $+0.15\%$ improvement (from 69.57% to 69.72% in Table S4).

Method	CIFAR100			iNat2018			Places-LT			Avg
	Head	Med	Tail	Head	Med	Tail	Head	Med	Tail	
LoRA	91.2	87.9	86.8	55.8	64.0	64.7	47.5	47.8	46.1	66.5
AdaptFormer	92.3	88.3	86.8	68.7	76.7	78.0	49.1	47.9	45.1	71.1
LoRA+AdaptFormer	92.5	88.4	87.0	67.9	77.2	77.8	48.6	48.0	45.3	71.2

Table S6. **Accuracy (%) of combining different additive PEFT methods.**

We also compare the ensemble of different PEFT methods in Table S6. Results show that simply combining LoRA and AdaptFormer yields marginal improvements ($+0.1\%$ on average) over using AdaptFormer alone, and shows the little gain for tail classes (*e.g.*, 77.8% vs 78.0% on iNat2018). This suggests that a naive or straightforward combination of PEFT methods is ineffective.

C.3. Hyperparameter Optimization is Particularly Crucial for Tail Classes

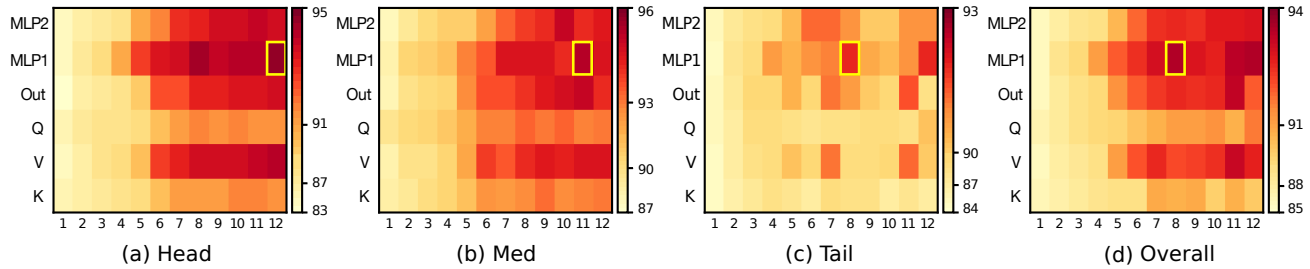

Figure S1. **Accuracy heatmap for head/med/tail/overall classes.** Visualization of accuracy heatmaps on different intra-block layers vs among different blocks (depth), where the block stands for the attention block of ViT. Results are reported on transfer scenarios IN21K→DOTA. The optima in each heatmap is highlighted by a yellow box. Results show that: 1) the (c) Tail classes exhibit non-monotonic accuracy changes across positions, while (a) Head and (b) Medium classes show monotonic trends; 2) the optimal configuration in (c) determines the overall optimal configuration in (d), indicating that tail-class performance dominates the model's overall performance. These findings validate our position-aware optimization strategy for long-tailed datasets.

The heatmaps in Figure 1 present the average accuracy across all classes. In this section, we provide a detailed breakdown of head/medium/tail/overall class performance for each grid search configuration in Figure S1.

Our analysis reveals that hyperparameter optimization is particularly challenging yet crucial for tail classes. First, for head classes (Figure S1(a)) and medium classes (Figure S1(b)), the accuracy shows a monotonic increase pattern towards deeper layers (8-12) and upper positions (MLP1/MLP2). In contrast, for tail classes, the accuracy values do not show a clear correlation with either depth or position (Figure S1(c)). Second, the performance on tail classes determines the model's overall performance: the optimal configuration (MLP1 position and depth 8) in Figure S1(c) directly corresponds to the overall optimal configuration in Figure S1(d). These findings highlight the importance of our optimization method for long-tailed datasets.

C.4. Analysis of LoRA Rank: Rank Optimization Can be Decoupled from Other Hyperparameters

We used fixed ranks in experiments (Section 4.3) because our method generalizes across different rank settings. We validate this claim through two aspects: 1) rank's impact on model performance, and 2) consistency of position/scaling factor observations across different ranks (Figure S2).

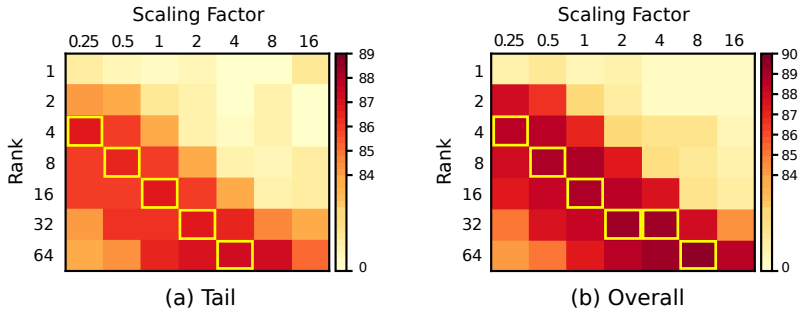

Figure S2. **Linear relationship between rank and optimal scaling factor.** The heatmaps show accuracy distributions across different ranks and scaling factors for (a) tail classes and (b) all classes. Results reported on IN21K→CIFAR100. Yellow highlights indicate the optimal scaling factor for each rank (the rank 32 in Figure (b) has two equally optima). We can observe a consistent linear relationship, *i.e.*, optimal scaling factor $\propto$ rank. This pattern suggests that rank optimization can be decoupled from remaining hyperparameters.

Concretely, from Figure S2, we have two key findings: 1) Rank selection significantly impacts model performance. For instance, in Figure S2(b), at scaling factor 4, changing the rank from 1 to 32 improves accuracy from 1.0% to 89.2% (dramatically +88.2%). 2) Rank and optimal scaling factor has a linear relationship. Specifically, the optimal scaling factor can be directly computed as $\frac{\text{rank}}{n}$, where n is a dataset-specific constant (*e.g.*, n=16 for CIFAR100-IR100 in Figure S2). This linear relationship suggests that the optimization of rank and remaining hyperparameters can be decoupled. Given these observations, our `MetaPEFT` method optimizes position and scaling factor hyperparameters while keeping the rank fixed.

C.5. t-SNE Feature Visualization

We provide two sample figures: t-SNE maps for (a) LoRA and (b) LoRA + Ours. LoRA + Ours achieves. We merge head categories and tail categories for clearer visualiztaion, and blue/red circles are head/tail classes, respectively. From the figure, we could observe better class separation and more compact cluster in each class.

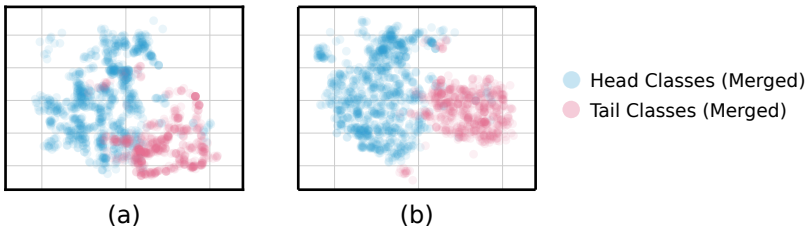

Figure S3. **t-SNE Visualization.** of (a) LoRA and (b) LoRA + ours.

C.6. Apply to More LoRA/Ensembled Methods.

The Table 6 of LIFT [5] shows that AdaptFormer and LoRA outperform other PEFT methods in long-tailed data distributions, especially for tail classes, so we adopt AdaptFormer/Adapter/LoRA as our baseline methods. Table 4 shows we beat LoRA by 1.2% for tail classes and 1.13% on average. Additionally, in Table S7, we apply our meta-optimization strategy to LIFT and recent low-rank methods LoTR [2] on IN21K→DOTA. Our method shows limited improvement on LIFT, which is expected as LIFT is already an excellent work with well-tuned scaling factors in its design.

D. Limitations and Future Works.

Despite the promising results of `MetaPEFT`, future works remain. First, its scalability to larger models and more diverse RS spectrum datasets warrants further investigation. Currently, evaluation is limited to the FUSRS SAR dataset due to the scarcity of SAR data, highlighting the need for larger, standardized SAR recognition benchmarks. Second, the current framework relies on a fixed backbone architecture (*e.g.*, ViT-B/16). Exploring how `MetaPEFT` generalizes across different backbone architectures, such as CNN models or hybrid transformer-CNN models, can broaden its applicability. Third, our bi-level optimization framework reduces overfitting in tail classes, however, it introduces additional computational overhead

Method	IN21K → DOTA		
	Head	Med	Tail
LIFT [5]	93.1	94.9	90.2
w/ Ours	93.2	94.7	90.6
LoTR [2]	93.0	93.1	90.3
w/ Ours	93.1	92.9	91.1

Table S7. **Apply our method to LIFT and LoTR.**

in outer loops. Fourth, our bi-level optimization framework still depends on several outer loop hyperparameters, including meta-parameter learning rate, learning steps, early stop, and update frequency. Reducing them remains an important direction for future research. Our future work could investigate lightweight optimization techniques or meta-learning strategies that reduce computational costs without compromising model performance.

As part of our ongoing next work, some more interesting exploations (*e.g.*, meta optimization under noisy data and insufficient data challenges) are currently ongoing. We will share our progress on our GitHub repository (link provided in the abstract).

References

[1] Sanjeev Arora, Nadav Cohen, Noah Golowich, and Wei Hu. A convergence analysis of gradient descent for deep linear neural networks. *arXiv preprint arXiv:1810.02281*, 2018. 2

[2] Daniel Bershatsky, Daria Cherniuk, Talgat Daulbaev, Aleksandr Mikhalev, and Ivan Oseledets. Lotr: Low tensor rank weight adaptation. *arXiv preprint arXiv:2402.01376*, 2024. 6, 7

[3] Yaoyao Liu, Bernt Schiele, and Qianru Sun. Adaptive aggregation networks for class-incremental learning. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, pages 2544–2553, 2021. 2

[4] Yaoyao Liu, Yingying Li, Bernt Schiele, and Qianru Sun. Online hyperparameter optimization for class-incremental learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 8906–8913, 2023. 2

[5] Jiang-Xin Shi, Tong Wei, Zhi Zhou, Jie-Jing Shao, Xin-Yan Han, and Yu-Feng Li. Long-tail learning with foundation model: Heavy fine-tuning hurts. In *Forty-first International Conference on Machine Learning*, 2024. 3, 6, 7

[6] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, 2017. 3

[7] Ross Wightman, Hugo Touvron, and Hervé Jégou. Resnet strikes back: An improved training procedure in timm. *arXiv preprint arXiv:2110.00476*, 2021. 3