Meta-Learning Hyperparameters for Parameter Efficient Fine-Tuning

Zichen Tian[1] Yaoyao Liu[2] Qianru Sun[1]

[1]Singapore Management University
[2]University of Illinois Urbana-Champaign

zichen.tian.2023@phdcs.smu.edu.sg, lyy@illinois.edu, qianru.sun@smu.edu.sg

Abstract

Training large foundation models from scratch for domain-specific applications is almost impossible due to data limits and long-tailed distributions — taking remote sensing (RS) as an example. Fine-tuning natural image pre-trained models on RS images is a straightforward solution. To reduce computational costs and improve performance on tail classes, existing methods apply parameter-efficient fine-tuning (PEFT) techniques, such as LoRA and AdaptFormer. However, we observe that fixed hyperparameters — such as intra-layer positions, layer depth, and scaling factors, can considerably hinder PEFT performance, as fine-tuning on RS images proves highly sensitive to these settings. To address this, we propose `MetaPEFT`*, a method incorporating adaptive scalers that dynamically adjust module influence during fine-tuning.* `MetaPEFT` *dynamically adjusts three key factors of PEFT on RS images: module insertion, layer selection, and module-wise learning rates, which collectively control the influence of PEFT modules across the network. We conduct extensive experiments on three transfer-learning scenarios and five datasets in both RS and natural image domains. The results show that* `MetaPEFT` *achieves state-of-the-art performance in cross-spectral adaptation, requiring only a small amount of trainable parameters and improving tail-class accuracy significantly.* [1]

1. Introduction

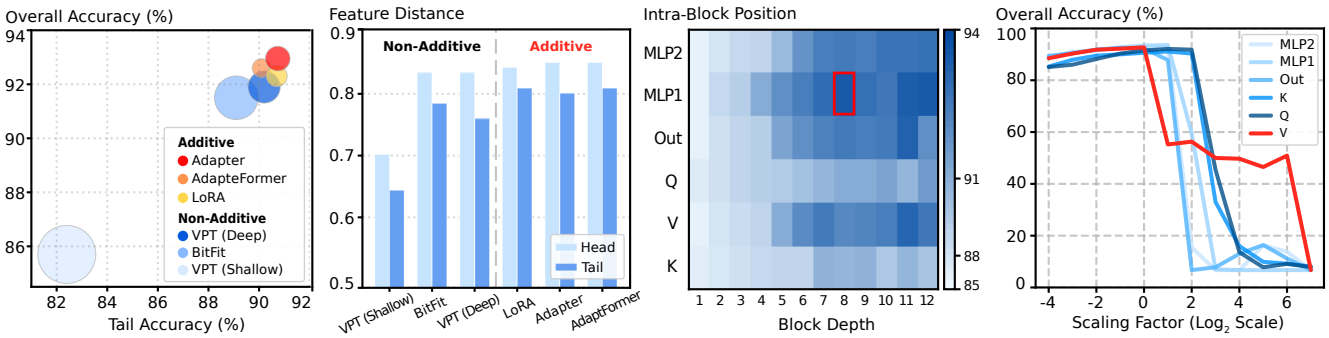
Training large foundation models from scratch for domain-specific applications presents fundamental challenges, particularly in domains with limited data availability and severe data imbalance issues. Take the remote sensing (RS) domain as an example. RS image recognition has critical applications in environmental monitoring, resource management, and disaster response [40, 58, 63]. However, learning large foundation models (*e.g.*, CLIP [43] and Stable Diffusion [44]) on RS data presents three fundamental challenges. First, RS data exhibits high spectral diversity, encompassing different spectral bands from optical remote sensing (ORS, 400-700nm) to synthetic aperture radar (SAR, 1mm-1m), which capture distinct physical properties of Earth's surface [14, 46]. This spectral diversity poses challenges for developing generic, cross-band models [21, 52]. Second, although large foundation models are prevalent in natural image domains, it is impractical to train dedicated foundation models for each RS spectral band due to data scarcity [14, 18, 64] and the prohibitive computational resources required for training or fine-tuning [7]. Third, RS data often displays a long-tailed distribution [6, 32, 54, 60], which causes fine-tuned foundation models to overfit to limited training samples in tail classes, resulting in poor generalization [50].

Adapting models from natural image domains to RS domains has become a more feasible and practical approach than training models from scratch. Compared to the resource-intensive process of full fine-tuning, Parameter-Efficient Fine-Tuning (PEFT) has emerged as a more efficient adaptation method for foundation models. PEFT includes various techniques, such as LoRA [25], AdaptFormer [5], and BitFit [57]. The common idea is to update only a small subset of parameters while keeping the pre-trained weights of the foundation models frozen. This approach reduces computational and data requirements, preserves general knowledge learned in natural image domains [55], and reduces overfitting in tail classes [47, 50].

In this paper, we conduct a comprehensive study of various PEFT methods in the RS domain and propose our improved method called `MetaPEFT` that is generic and more efficient than any individual technique of PEFT. Specifically, we study five representative PEFT methods:[2] a selective method called BitFit [57]; three additive methods respectively called LoRA [25], Adapters [23], and Adapt-

[1] Code: https://github.com/doem97/metalora

[2]We provide detailed PEFT taxonomy and our classification criteria of LoRA in Section 2.



(a) Compare PEFT Performance

(b) Inter-Class Feature Distance

(c) Intra-Block Position v.s. Block Depth

(d) Intra-Block Position v.s. Scaling Factor

Figure 1. **Comparing PEFT methods for the model adaptation of IN21K → DOTA.** (a) Bubble plot of overall accuracy, tail-class accuracy, and performance variance (in standard deviation) of 5 PEFT methods in 6 total versions. Additive methods exhibit consistently higher accuracy and lower variance than non-additive methods. (b) Inter-class feature distances of the PEFT methods measured by cosine similarity. Additive methods achieve 13% further feature distances (which means better discrimination among tail classes) with comparable head-class distances. (c) Accuracy heatmap for applying PEFT on different positions of ViT: on different intra-block layers v.s. among different attention blocks (depth). Deeper blocks yield better performance (86.5% to 90.4%), but the combination of optimal block and intra-block position shows unexpected degradation (0.6% drop for FFN layer with depth 10→11). The ▯ marks optimal combination. (d) Accuracy heatmap of intra-block positions v.s. scaling factors. Different positions show distinct sensitivity to scaling factors, with sharp accuracy drops observed (*e.g.*, applying PEFT on the attention output layer (denoted as Out) drops from 87% to 6.7% when its PEFT scaling factor increases from 2 to 4). These highlight the non-monotonic complexity of PEFT hyperparameters.

Former [5]; and a soft-prompt method called VPT [27]. We make two key observations. First, additive methods outperform non-additive methods in both performance and stability. As shown in Figure 1(a), additive methods consistently achieve higher overall accuracy and tail-class accuracy while maintaining lower performance variance (represented by bubble size). Similar observations in the NLP domain [24] and natural image domain [20] suggest that such superiority stems from both architectural design (zero-initialization) and hyperparameter choice (scaling factor). Zero-initialization ensures parameter updating starts from the pre-trained state, while the scaling factor preserves update directions and modulates only magnitudes. Second, additive methods achieve better tail-class feature discrimination. As visualized in Figure 1(b), additive methods achieve 13% higher inter-class feature distances by average (using cosine distance) for tail classes while maintaining comparable distances for head classes. We attribute this improvement to its flexibility in insertion positions. For example, the soft-prompt method VPN operates at only input layers. In contrast, additive modules like LoRA and AdaptFormer can be inserted at diverse positions (*e.g.*, across attention blocks of ViT and on the Q/K/V [25] or FFN layers [5] of any attention block). Such flexibility enables effective model adaptation without needing large-scale training data, benefitting tail classes [29, 56]. We thus use additive PEFT as a strong baseline to tackle RS tasks.

We found that additive methods vary mainly based on three key hyperparameters: adaptation position inside an individual attention block, adaptation position across attention blocks (depth), and scaling factors.[3] For example, LoRA applies its low-rank adaptation to Q/K/V layers, while AdaptFormer applies its adaptation weights to FFN layers. We conduct thorough ablation studies on these three hyperparameters and present the resulting heatmap visualizations in Figures 1(c) and 1(d). From the figures, we make two key observations. 1) Model performance exhibits high sensitivity to both hyperparameters. Specifically, accuracy varies by 86% across scaling factors (from 8.1% to 91.1% on the K projection layer), 4.0% across attention block depths (from 86.5% in block 1 to 90.5% in block 11), and 2.4% across intra-block positions (from 90.4% on K projection layer to 92.8% on FFN layer). 2) Individual hyperparameters demonstrate monotonic trends, *e.g.*, deeper insertion positions generally lead to better performance in Figure 1(c). Their combinations show complex non-monotonic effects in Figure 1(d). In other words, combining individually optimal hyperparameters often leads to unexpected performance degradation, *e.g.*, using optimal intra-block position (*i.e.*, FFN) with optimal block depth (*i.e.*, depth 11) results in a 0.6% accuracy drop. Although an exhaustive search might handle these non-monotonic effects, it is computationally infeasible due to the large configuration space of complexity $\mathcal{O}(L|\mathcal{S}|N_\alpha)$ (Section 3.2). Moreover, gradient-based optimization is infeasible as PEFT hyperparameter tuning poses a mixed discrete-continuous optimiza-

[3] Other relevant but less significant hyperparameters, such as module size and initialization, will be discussed in the Appendix.

tion problem: jointly optimizing discrete positions and continuous scaling factors (elaborated in Section 3.2). These challenges motivate us to develop an efficient end-to-end hyperparameter optimization method for PEFT.

Specifically, we propose a meta-learning method called `MetaPEFT`, by introducing two key designs: 1) a unified modulator of three PEFT-specific hyperparameters (*i.e.*, intra-attention-block position and attention-block depth on ViT, and the scaling factor of PEFT) and 2) a bi-level optimization framework to learn this modulator without needing additional training data. First, the unified modulator contains a set of scalars, each applied to an individual PEFT position (*e.g.*, a Q/K/V projection layer, attention output layer (denoted as Out), or an FFN layer in the attention block of ViT-B/16), without introducing large overhead (*e.g.*, less than 800 additional parameters for LoRA on ViT-B/16). This modulator controls both PEFT's insertion position and scaling factor: when the value of a scalar is close to zero, the PEFT is deactivated at the corresponding position (of the scalar); when this value is significantly high, its magnitude determines "how strong PEFT is for the adaptation". Introducing this modulator makes gradient-based optimization possible due to its differentiable nature. Second, we optimize this modulator through a bi-level optimization framework with two alternating loops: the inner loop trains the model parameters with the modulator (*i.e.*, all scalars) fixed, and the outer loop optimizes the modulator on a randomly sampled subset of training data in each iteration. This dynamic sampling exposes optimization to diverse data subsets. It hence prevents overfitting, especially effective for tail classes (*e.g.*, 3.5 times higher accuracy improvement for tail classes than head classes on the setting of "SatMAE $\rightarrow$ SAR", with LoRA as a baseline). Empirically, we found that direct optimization often leads to numerical instability (*i.e.*, negative values); thus, we apply softplus activation to constrain the modulator to be non-negative. In summary, our `MetaPEFT` has three key advantages: 1) it converts the mixed discrete-continuous optimization problem into a differentiable formulation, thereby allowing gradient-descent-based optimization; 2) it automatically discovers the optimal adaptation strength for each position; and 3) its dynamic sampling of meta-learning samples (*i.e.*, validation samples) greatly reduces model overfitting to tail classes.

2. Related Works

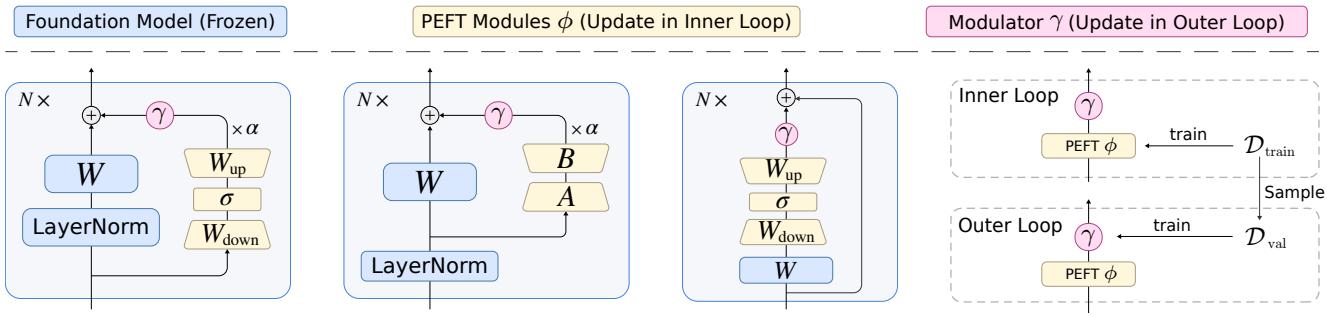
Long-tailed Model Adaptation. Long-tailed learning approaches in computer vision can be categorized into three main strategies: data manipulation through re-sampling and re-weighting techniques exemplified by cRT [29] and BBN [62], representation learning via contrastive learning as demonstrated by PaCo [8], and classification objective modification methods such as LDAM [3] and Focal Loss [45]. However, training from scratch often yields suboptimal results compared to leveraging pre-trained models [56]. Recent works explore foundation models like CLIP, where BALLAD [41] and VL-LTR [49] directly fine-tune the entire model, while LPT [10] incorporates external knowledge. Despite promising results, they face fundamental trade-offs between performance and efficiency [47, 59].

Parameter-Efficient Fine-tuning (PEFT). PEFT enables efficient model adaptation while avoiding the computational overhead of full fine-tuning [15, 23]. By modifying only a small subset of parameters, PEFT achieves comparable performance with significantly reduced costs [20]. Current approaches include additive methods such as adapters [23] and LoRA [25], and selective methods BifFit [57] or DiffPrune [17]. However, these methods are highly sensitive to hyperparameter choices, including LoRA's rank, adapter dimensions, and selection criteria [33, 47]. Our empirical study reveals that while individual hyperparameters show monotonic trends, their combinations lead to complex, non-monotonic effects. This makes manual optimization both time-consuming and suboptimal, especially given the mixed discrete-continuous nature of these hyperparameters [4, 47].

Meta-Learning for Hyperparameter Optimization. Traditional hyperparameter optimization through random search [2] or Bayesian optimization [48] struggles with PEFT scenarios, as they ignore structural relationships between network modules and require expensive sequential optimization. While methods like population-based training [26] and BOHB [11] attempt parallel optimization, they remain computationally intensive. Meta-learning approaches such as MAML [12] and Meta-SGD [35] demonstrate effective learning-to-learn strategies, with online meta-learning [13, 36–38] enabling continuous adaptation. Neural architecture optimization methods, including DARTS [19] and Auto-Meta [30], successfully apply meta-learning principles. However, existing approaches focus primarily on general hyperparameters [1] or architecture choices [65] without addressing PEFT-specific challenges. The complex interdependencies in PEFT methods [20, 24, 25] create a mixed discrete-continuous optimization problem requiring specialized solutions.

3. Method

We have made two key observations in Section 1: 1) additive PEFT methods excel at tail classes, and 2) their hyperparameters exhibit complex non-monotonic combination effects. In Section 3.1, we give a general formulation for additive PEFT methods and elaborate on their specific hyperparameters, such as discrete layer positions and continuous scaling factors. The results of PEFT in Figure 1(c)-(d) show high sensitivity to the values of hyperparameters. We thus explore hyperparameter optimization. In Section 3.2, we formalize the manual hyperparameter optimization as a mixed integer non-linear programming (MINLP) problem, which is impractical to solve for deep learning models.



(a) AdaptFormer w/ Ours (b) LoRA w/ Ours (c) Adapter w/ Ours (d) Optimization Framework

Figure 2. **Architecture and optimization framework for `MetaPEFT`. In Figures (a)-(c)**, we illustrate how our proposed modulator γ is integrated with three representative additive PEFT methods: (a) AdaptFormer with modulated up/down projections (W_{up}/W_{down}), (b) LoRA with modulated low-rank decomposition matrices (A/B), and (c) Adapter with modulated projection layers (W_{up}/W_{down}). The σ denotes the non-linear activation function (*e.g.*, ReLU). **In Figure (d)**, we show our bi-level optimization framework. The inner loop optimizes PEFT parameters ϕ on training data $\mathcal{D}_{\text{train}}$, and the outer loop updates modulator γ on validation data $\mathcal{D}_{\text{val}}$ sampled from the training set. The symbol $N\times$ indicates the operation is repeated for N attention blocks (*e.g.*, N=12 for `ViT-B/16`).

In Section 3.3, we propose our approach, `MetaPEFT`. We unify both discrete and continuous hyperparameters by converting them into scalars, each applied independently to a specific network position, and optimize them end-to-endly.

3.1. PEFT and Its Hyperparameters

Parameter Efficient Fine-Tuning (PEFT). Given a pre-trained model parameterized by θ, PEFT transfers pre-trained representations by updating only a small set of parameters ϕ, *i.e.*, its dimension is usually $\dim(\phi) \ll \dim(\theta)$:

$$y = f(x; \theta, \phi), \tag{1}$$

where x is the input and y is the output representation. θ is kept frozen during updating, and only ϕ is updated. In this work, we focus on additive PEFT methods for their high effectiveness (clarified in Section 1). While non-additive methods modify fixed parameters in the pre-trained model (*i.e.*, $\phi \in \theta$), additive methods introduce new parameters ϕ to carefully selected positions in the model:

$$y = f(x; \theta) + \mathbb{1}_p(\alpha \cdot \Delta(x; \phi)), \tag{2}$$

where $x \in \mathbb{R}^d$ is the intermediate feature at position p, $f(\cdot)$ is the original layer operation (*e.g.*, linear transformation, attention projections or MLP) with parameters θ, $\Delta(x; \phi)$ is the additive PEFT module parameterized by ϕ, α is the scaling factor, and $\mathbb{1}_p \in \{0, 1\}$ indicates whether the PEFT module is used (1) or not (0) at position p. This formulation reveals why additive PEFT achieves improved overall performance and tail-class feature discrimination (Section 1): $\mathbb{1}_p$ allows adaptation at crucial positions, while α precisely controls the adaptation strength, enabling effective feature adjustment, especially for data-scarce tail classes.

Fomula 2 is for general additive PEFT methods. Different methods have distinct $\Delta(x; \phi)$ designs, as shown in Figure 2 (a)-(c) in yellow color. For example, LoRA [25] introduces low-rank decomposition of weight W (*i.e.*, $\phi = \{A, B\}$), which can be formulated as:

$$y = (x \cdot W) + \mathbb{1}_p(\alpha \cdot x \cdot BA^\top), \tag{3}$$

and Adapter/AdaptFormer introduce up/down projections (*i.e.*, $\phi = \{W_{\text{down}}, W_{\text{up}}\}$) and activation σ, formulated as:

$$y = f(x; \theta) + \mathbb{1}_p\left(\alpha \cdot W_{\text{up}} \cdot \sigma\left(W_{\text{down}} \cdot x\right)\right). \tag{4}$$

Block Depth d and Intra-block Position s. In transformer architectures, the position p in Formula 2 consists of two discrete dimensions: block depth $d \in \{1, ..., L\}$ and intra-block position $s \in S$, where L is the total number of layers (*e.g.*, 12 for ViT-B/16) and $\mathcal{S}$ is the set of insertion positions within a transformer block (*i.e.*, Q/K/V projection layers, attention output layer, and FFN layer). Therefore, $\mathbb{1}_p$ can be also written as $\mathbb{1}_{d,s}$. Deeper d and s generally yield higher accuracy with improved tail-class separation (Figure 1(c)). However, these discrete parameters cannot be optimized through gradient descent, yet manual hyper-parameter optimization often leads to unexpected performance drops (Figure 1(c)). In the following subsections, we will introduce an equivalent way of optimizing them but in an end-to-end manner.

Scaling Factor α. The scaling factor α in Formula 2 is a continuous hyperparameter with a range of $(0, \infty)$. Empirically, α is usually set within $[0.001, 1]$ or start searching from 0.1 [5]. It is a hyperparameter used in the forward process of the model training and controls the weight of adding the PEFT representations to the original pre-trained features. In the Appendix, we justify its relationship with the learning rate. Please note that we do not meta-learn the learning rate as they are not specific to PEFT.

3.2. Limitations of Manual Optimization

Our empirical study (Section 1) has shown that the joint optimization of hyperparameters exhibits complex, non-monotonic interactions. This makes heuristic or exhaustive search inefficient and challenging. In this section, we further theoretically justify that this optimization objective is intractable, and identify its core challenge as a mixed discrete-continuous (MINLP) problem.

Optimization Objective. Manual optimization performs exhaustive evaluations of configurations. Its optimization objective can be formulated as follows:

$$\begin{aligned} \max_{d,s,\alpha} \quad & \mathcal{A}_{\text{val}}(\phi^*_{d,s,\alpha}; \mathcal{D}_{\text{val}}), \\ \text{s.t.} \quad & d \in \{1, ..., L\}, s \in \mathcal{S}, \alpha > 0, \\ & \phi^*_{d,s,\alpha} = \arg\min_{\phi} \mathcal{L}_{\text{train}}(\phi, d, s, \alpha; \mathcal{D}_{\text{train}}), \end{aligned} \tag{5}$$

where $\phi^*_{d,s,\alpha}$ denotes the optimal PEFT parameters under the given hyperparameter configuration $\{d, s, \alpha\}$. $\mathcal{A}_{\text{val}}$ denotes the validation accuracy, and $\mathcal{L}_{\text{train}}$ denotes the training loss (Logit Adjustment [42] loss, in our case).

MINLP Problem. The optimization objective fomula 5 forms a Mixed Integer Non-Linear Programming (MINLP) problem. This poses two key challenges. First, the mix of continuous variable α and discrete variables d, s makes this problem non-differentiable, preventing the direct use of gradient-descent-based optimization methods. We also prove this problem is NP-hard in the Appendix. Second, the problem has a large discrete configuration space, *i.e.*, $\mathcal{O}(L|\mathcal{S}|N_\alpha)$. This makes exhaustive search approaches computationally infeasible, *e.g.*, manually optimizing ViT-B/16 (L=12, $|\mathcal{S}|$=5) with only N_α=10 requires exploring 600 configurations.

3.3. Auto Optimization via Meta-Learning

To address this mixed discrete-continuous challenge, we propose to unify all hyperparameters into a differentiable modulator that can be optimized through gradient.

Unified Modulator. We unify the positional indicator $\mathbb{1}_p$ and scaling factor α (in Equation 2) into a single differentiable modulator $\gamma \in \mathbb{R}$, as illustrated by pink in Figure 2:

$$y = f(x; \theta) + \gamma \cdot \Delta(x; \phi). \tag{6}$$

This design turns the MINLP problem (Equation 5) into a continuous optimization. Concretely, when $\gamma \approx 0$, it suppresses the PEFT module's contribution to zero, functionally equivalent to $\mathbb{1}_p = 0$; when $\gamma > 0$, it controls both module activation (*i.e.*, continuous relaxation of $\mathbb{1}_p$) and update magnitude (role of α). In practice, we initialize $\gamma = 1.0$ to preserve the model's pre-trained behavior during the first inner loop (note that while PEFT modules' zero-initialization preserves pre-trained behavior at the first training step, γ remains frozen during the first inner loop), and adopt softplus activation to ensure non-negativity and numerical stability of γ.

Bi-Level Optimization Framework. The continuous γ allows optimization through gradients on the validation set. We reformulate the optimization objective 5 as follows:

$$\begin{aligned} \min_{\gamma \in \mathbb{R}^+} \quad & \mathcal{L}_{\text{LA}}(\phi^*_\gamma; \mathcal{D}_{\text{val}}), \\ \text{s.t.} \quad & \phi^*_\gamma = \arg\min_{\phi} \mathcal{L}_{\text{LA}}(\phi, \gamma; \mathcal{D}_{\text{train}}), \end{aligned} \tag{7}$$

where $\mathcal{L}_{\text{LA}}$ denotes the Logit Adjustment [42] loss. We solve this bi-level optimization by alternating inner loop

$$\phi_{t+1} = \phi_t - \eta_\phi \nabla_\phi \mathcal{L}_{\text{LA}}(\phi_t, \gamma_t; \mathcal{D}_{\text{train}}) \tag{8}$$

and outer loop

$$\gamma_{t+1} = \gamma_t - \eta_\gamma \nabla_\gamma \mathcal{L}_{\text{LA}}(\phi_{t+1}; \mathcal{D}_{\text{val}}), \tag{9}$$

where η_ϕ and η_γ are learning rates for inner and outer loops, respectively. The inner loop optimizes PEFT parameters with frozen modulator γ, while the outer loop updates modulator γ with frozen PEFT parameters. In practice, for each outer loop, we randomly sample 20% of training data as $\mathcal{D}_{\text{val}}$. This sampling strategy serves as an implicit regularization, *i.e.*, it reduces hyperparameter overfitting by exposing the optimization to diverse data subsets. This strategy particularly benefits tail classes where overfitting is severe. Besides, we alternate between inner/outer loops every K inner steps (K is discussed in Section 4). This alternating strategy balances sufficient optimization of PEFT parameters while ensuring timely modulator updates.

4. Experiments

4.1. Model Adaptation Scenarios

To comprehensively evaluate our method's effectiveness in addressing core challenges in RS and natural image domains, we design experiments in three transfer-learning scenarios. These scenarios systematically validate our method's capability in handling spectral diversity, data scarcity, and long-tailed distribution issues in both RS and natural image domains.

Natural Vision Domain Adaptation. In the natural image domain, we use three standard long-tailed benchmarks: CIFAR100-IR100 [31] (50K samples, 100 classes, imbalance ratio 100), Places-LT [39, 61] (62K samples, 365 classes, imbalance ratio 996), and iNaturalist-2018 [51] (437.5K samples, 8,142 species, imbalance ratio 500). Each dataset has varying imbalance ratios, thus can help establish our method's baseline performance on conventional long-tailed benchmarks.

Natural Vision to RS Domain Adaptation. To evaluate our method's cross-domain adaptation and data scarcity handling abilities, we transfer an IN21K pre-trained ViT-B/16 model to one representative optical remote sensing (ORS) dataset, DOTA [9, 28, 53], which contains 188K images across 15 object categories. DOTA exhibits a significant imbalance ratio of 86, with class distribution (in the whole dataset) ranging from 28.35% (ship) to 0.64% (helicopter). Following standard practice, we categorize the classes into the head (6 classes, >5% samples), middle (3 classes, 1-5%), and tail (6 classes, <1%) groups.

Cross-Spectral RS Domain Adaptation. To validate our method's effectiveness in handling spectral diversity, we transfer the optical pre-trained model, SatMAE-L/16 (non-temporal edition) [7], to the synthetic aperture radar (SAR) domain using the FUSRS v2 dataset [50]. This dataset combines FUSAR-Ship [22] and SRSDD [34] datasets, containing 5,261 high-resolution (1m-10m) SAR ship images for training. We use categories with $< 20\%$ training samples as tail classes. The large domain gap between ORS and SAR, coupled with the small dataset and class imbalance, makes this adaptation scenario particularly challenging.

4.2. Implementation Details

Our `MetaPEFT` alternates between training PEFT modules and learning their modulators through a bi-level optimization framework. Here, we detail the implementation of both alternate steps.

PEFT Configuration. We adopt a principled approach [43] to determine PEFT module sizes based on dataset complexity. We control the size of the PEFT module based on setting up different ranks in LoRA or different adapter dimensions in Adapter/AdaptFormer. For simplicity, we use the term "PEFT dimension" in the following texts to represent the rank/dimension. For natural vision datasets, we set PEFT dimension proportional to the number of classes while maintaining parameter efficiency: rank 4 for CIFAR100-IR100 (100 classes), rank 8 for Places-LT (365 classes), and rank 256 for iNaturalist-2018 (8,142 classes). For RS datasets, considering their domain-specific characteristics and limited data, we empirically set the PEFT dimension as 4 after validation experiments showing little gains from using larger PEFT dimensions.

Meta Training Protocol. Our training follows the bi-level optimization framework elaborated in Section 3.3. The inner loop optimizes PEFT parameters using SGD with a base learning rate of $\eta_\phi = 1 \times 10^{-2}$ using a Logit Adjustment loss [42]. We use a base batch size 128 with square-root scaling [16] for learning rate adjustment. For the outer (meta) loop, we randomly hold out 20% of training data stratified by class to maintain class distribution. The outer loop optimizes our proposed modulator using Adam with the meta-learning rate η_γ (hyperparameter that varies depending on the dataset and backbone) and L2 regularization. It has the same loss function as the inner loop. Especially for RS datasets, we make specific adjustments: DOTA uses 20 inner loop steps with batch size 512, while FUSRS uses 10 steps with batch size 256 and a reduced learning rate due to limited data. We implement early stopping when validation accuracy shows less than 0.3% improvement over 3 consecutive epochs. All experiments are conducted on four NVIDIA V100/3090 GPUs, with training times ranging from 2 hours (CIFAR100-IR100) to 6 hours (iNaturalist-2018). More implementation details are available in the Appendix.

Fair Comparison. we ensure a fair comparison between low-rank and other additive methods. Concretely, we ensure a similar module size between LoRA and adapters by fixing the rank of LoRA and the adapter's hidden dimension to a comparable size. We follow the setting in [47] to set higher hidden dimensions for larger classifier heads. The learnable parameters in different methods on different datasets are provided in Appendix Table S2.

4.3. Ablation Studies and Method Comparisons

Our observations in Section 1 identify three key hyperparameters: intra-block position, block depth, and scaling factor. In this section, we provide detailed ablation results for each of them and their combinations.

Impact of intra-block position. We examine the effectiveness of different positions (projection layers) within an attention block. As shown in Table 2, the MLP1 (*i.e.*, the first FFN layer) achieves the best performance (93.4% average accuracy). It performs particularly better (than others) in tail classes (91.6%). The results also show that FFN positions (MLP1, MLP2) outperform attention-related positions (K, Q, V, Out), with performance gaps ranging from 0.7% to 2.8%. This might be because FFN layers focus on feature transformation, while attention layers primarily handle spatial correlations. This feature makes FFN layers more suitable for domain adaptation, where the distribution shift is the main challenge. Besides, in Table 1, we examine different layer combinations. Results show that attention and FFN modules are complementary to each other (*e.g.*, ATTN+FFN obtained 1.3% improvement on DOTA's tail classes). We think this complementarity is because attention focuses on global context and FFN processes local features. Combining them results in more robustness.

Impact of block depth. Figure 1(c) shows the impact of cumulative last-N blocks. As an extended investigation, in Table 3, we show the results of different block groups. The middle-lower blocks (L3-5) achieve the best performance (91.9%), followed closely by middle-upper blocks (L6-8). Intriguingly, the deepest blocks (L9-11) show the largest performance drop (3.2%). This effect is amplified for tail classes, where the gap between optimal and sub-

Position	CIFAR100			iNat2018			Places-LT			DOTA			Avg	Avg_{tail}
	Head	Med	Tail	Head	Med	Tail	Head	Med	Tail	Head	Med	Tail		
ATTN	92.2	87.7	86.3	**67.9**	76.3	76.7	48.4	48.3	**45.7**	94.1	94.7	91.1	75.8	74.9
FFN	**92.4**	87.9	**86.8**	66.7	75.6	76.9	**48.6**	47.9	45.6	**94.6**	94.8	91.4	75.8	75.2
ATTN-FFN	92.3	**88.1**	86.6	67.7	**76.8**	**77.5**	48.2	**48.3**	45.5	**94.6**	**95.4**	**92.4**	**76.1**	**75.5**

Table 1. **Applying PEFT on Different Intra-Block Layer Positions.** We compare three LoRA insertion positions within the attention block: attention-projection-layers-only (ATTN), feedforward-network-only (FFN), and combined (ATTN-FFN) across four adaptation settings: IN21K pre-trained ViT-B/16 $\rightarrow$ {CIFAR100, iNaturalist2018, Places-LT, DOTA}. Results reported in Accuracy (%). Results show that combined positions yield slightly better overall performance and tail-class accuracy.

Module	Position	Accuracy (%)			
		Head	Med	Tail	Avg
ATTN	K	91.6	93.0	87.7	90.6
	Q	91.4	93.1	90.0	91.5
	V	94.0	94.5	90.2	92.7
	Out	93.6	94.1	90.0	92.3
FFN	MLP 1	**94.6**	**94.6**	**91.6**	**93.4**
	MLP 2	93.6	94.3	90.6	92.7

Table 2. **Applying PEFT on Layer-Wise Intra-Block Positions for IN21K$\rightarrow$DOTA.** We compare LoRA's different layer-wise insertion positions within the attention block. ATTN in Table 1 now has 4 versions due to 4 kinds of projection layers (Q/K/V/Out) in the attention computation&output of the attention block. Results reported in Accuracy (%). LoRA on the first FFN layer (MLP 1) yields the best performance with an average accuracy of 93.4%.

Block Group	Many	Med	Few	Avg	Drop (%)
L9-11	88.1	89.8	88.8	89.0	3.2
L6-8	92.3	93.5	89.3	91.6	0.3
L3-5	**92.4**	**93.7**	**89.8**	**91.9**	-
L0-2	90.5	92.8	88.4	90.6	1.4

Table 3. **Ablate Different Block Depths for IN21K$\rightarrow$DOTA (%).** We analyze the impact of applying LoRA across different transformer blocks. Results reported in Accuracy (%). Results indicate that using LoRA on the middle-lower blocks (L3-5) achieves the best performance (91.9% overall accuracy), while the deepest blocks (L9-11) show an unexpected performance drop. This suggests that we should apply a higher scaling factor for LoRA in intermediate blocks.

optimal block reaches 4.5%. These observations suggest PEFT modules should be applied with a stronger scaling factor on middle blocks rather than being applied uniformly or only to deep blocks. In addition, we also visualized the effect of using joint hyperparameters in Figure 1. We will provide more detailed results (including head/med/tail performances) in the supplementary materials.

Random sampling strategy. We study how the proportion of training data in the outer loop impacts performance. As shown in Table 6, increasing the sampling ratio from 10% to 30% leads to consistent performance improvements across all classes, with the most significant gains observed in tail classes (from 90.8% to 93.4%). This trend suggests that insufficient sampling may lead to sub-optimal modulator optimization. We adopt 20% as our default setting in main comparisons, as it provides a good balance between performance (94.2% average accuracy) and training time.

Compare with state-of-the-art. We evaluate our method by incorporating it into three representative additive PEFT methods. Results in Table 4 reveal three key findings. *First*, our method shows strong synergy with LoRA, consistently improving its performance across all metrics and achieving the highest average accuracy (+1.13% to 83.97% for LoRA). *Second*, our method demonstrates particular strength in handling large domain gaps. In the challenging SatMAE$\rightarrow$SAR scenario, where the domain shift is most significant, our method achieves its largest improvement on tail classes (+1.2% for LoRA). This potentially stems from our method's ability to selectively strengthen or weaken different blocks' adaptation based on their domain-specific importance. *Third*, the consistent improvements over iNat2018 (8,142 classes) demonstrate our method's capability to tackle extreme long-tailed scenarios. Even with such extensive class space, our method still provides stable gains (+0.8% on tail classes). This is because our random sampling strategy forces the model to optimize the modulator on diverse subsets of data in each iteration, reducing model overfitting to some extent.

"Are we learning a better representation?" Feature distance analysis in Table 5 unveils why our method can improve the tail-class performance. First, all additive methods show higher inter-class distances than non-additive ones, the gap is particularly pronounced for tail classes (0.77-0.78 vs. 0.72). Second, when combined with LoRA, our method achieves the best feature separation (0.85 for the head, 0.82 for the tail) and reduces the performance gap between head and tail class groups (from 0.04 to 0.03). This balanced improvement stems from our dynamic modulation mechanism, which automatically adjusts adaptation

Category	Method	IN21K → iNat2018			IN21K → DOTA			SatMAE → SAR		Avg_{tail}	Avg
		Head	Med	Tail	Head	Med	Tail	Head	Tail		
Non-additive	VPT-Shallow	57.5	65.5	65.9	85.4	89.0	82.4	39.8	68.4	72.23	69.24
	BitFit	57.6	67.3	68.4	92.0	93.7	89.1	32.1	74.7	77.40	71.86
	VPT-Deep	64.9	74.2	75.9	92.1	93.5	90.2	30.6	50.0	72.03	71.43
Additive	Adapter	67.9	76.8	77.7	93.2	94.9	90.6	37.9	75.8	81.37	76.85
	w/ Ours	68.2	76.9	78.1	93.2	**95.1**	90.7	37.7	76.0	81.60	76.99
	AdaptFormer	68.7	76.7	78.0	93.2	94.9	90.1	35.3	**76.7**	81.60	76.70
	w/ Ours	68.7	76.6	78.2	92.7	94.6	90.1	35.5	76.4	81.57	76.60
	LoRA	69.1	77.3	78.5	93.1	93.4	90.7	40.0	72.1	80.43	76.78
	w/ Ours	**70.2**	**78.6**	**79.3**	**93.9**	**95.1**	**91.4**	**40.6**	74.2	**81.63**	**77.91**

Table 4. **Comparing different PEFT methods *w/* or *w/o* ours.** We evaluate three transfer scenarios (IN21K→iNat2018, IN21K→DOTA, SatMAE→SAR) using non-additive methods (VPT, BitFit) and additive methods (Adapter, LoRA, AdaptFormer). Performance is measured across head, medium, and tail classes for each dataset (the SAR dataset has only head and tail) and reported in Accuracy (%). "Avg_{tail}" shows mean performance on tail classes, and "Avg" is the macro-average across all class splits. Results show that our method consistently improves additive methods, with LoRA achieving the highest tail-class performance (81.63%). Ours are marked in gray.

Category	Method	Inter-class Distance	
		Head	Tail
Non-additive	VPT-Shallow	0.65	0.58
	BitFit	0.81	0.75
	VPT-Deep	0.81	0.72
Additive	Adapter	0.83	0.77
	w/ Ours	0.84	0.79
	AdaptFormer	0.83	0.78
	w/ Ours	0.83	0.77
	LoRA	0.82	0.78
	w/ Ours	**0.83**	**0.80**

Table 5. **Inter-Class Feature Distance.** We measure the inter-class *cosine* distance between head classes, and between tail classes. Results are reported on transfer scenario IN21K→DOTA using average cosine distance, where higher values indicate better class separation. Our method (marked in gray) achieves the best feature separation when combined with LoRA.

strength across different intra-block positions and block depths. Finally, the low distance of VPT-Shallow confirms our assumption that input-only modifications are insufficient for complex domain adaptation.

Impact of rank in low-rank methods. In Table 4, we found rank r one critical hyperparameter for low-rank methods (*e.g.*, LoRA and its variants). It significantly affects the performance and size of PEFT modules. Our experiments also demonstrate this rank r is independent of scaler and positions. We provide detailed observations about rank selection in the Appendix.

Computational overhead. Our `MetaPEFT` archives 1.13 percentage points higher accuracy than LoRA (Table 4) with minimal overhead (*e.g.* only 0.0008M params on ViT-B/16). The detailed computational overhead is provided in the Appendix.

Ratio	Head	Med	Tail	Avg	ΔTail (%)
10%	93.4	94.2	90.8	92.8	-2.6
20%	94.7	95.4	92.5	94.2	-1.0
30%	**95.0**	**95.8**	**93.4**	**94.7**	-

Table 6. **Impact of Sampling Strategies (%).** We evaluate different sampling ratios in the outer loop of `MetaPEFT`. Results reported in Accuracy (%). While 30% sampling achieves the best performance (94.7% average accuracy), we adopt 20% in our default setting due to time constraints.

5. Conclusions

In this paper, we identified that additive PEFT methods outperform non-additive ones, but their performance is highly sensitive to insertion positions and scaling factors. We extensively evaluated this phenomenon on five transfer learning scenarios in both RS and natural image domains. To address this challenge, we proposed MetaPEFT, a meta-learning framework that converts discrete and continuous hyperparameters into a unified differentiable modulator optimized through bi-level optimization. Our approach effectively handles the spectral diversity and long-tailed data distributions without requiring extensive manual tuning. We discuss the limitations and future work in the Appendix.

Acknowledgments

The authors gratefully acknowledge the support from the DSO research grant awarded by DSO National Laboratories, Singapore. The authors also extend sincere gratitude to Prof. Antoine Ledent (Singapore Management University) for his insightful guidance about the non-convex properties of PEFT optimization during the rebuttal.

References

[1] Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando De Freitas. Learning to learn by gradient descent by gradient descent. *Advances in neural information processing systems*, 29, 2016. 3

[2] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(2), 2012. 3

[3] Kaidi Cao, Colin Wei, Adrien Gaidon, Nikos Arechiga, and Tengyu Ma. Learning imbalanced datasets with label-distribution-aware margin loss. *Advances in neural information processing systems*, 32, 2019. 3

[4] Jiaao Chen, Aston Zhang, Xingjian Shi, Mu Li, Alex Smola, and Diyi Yang. Parameter-efficient fine-tuning design spaces. *arXiv preprint arXiv:2301.01821*, 2023. 3

[5] Shoufa Chen, Chongjian Ge, Zhan Tong, Jiangliu Wang, Yibing Song, Jue Wang, and Ping Luo. Adaptformer: Adapting vision transformers for scalable visual recognition. *Advances in Neural Information Processing Systems*, abs/2205.13535, 2022. 1, 2, 4

[6] Gordon Christie, Neil Fendley, James Wilson, and Ryan Mukherjee. Functional map of the world. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6172–6180, 2018. 1

[7] Yezhen Cong, Samar Khanna, Chenlin Meng, Patrick Liu, Erik Rozi, Yutong He, Marshall Burke, David Lobell, and Stefano Ermon. Satmae: Pre-training transformers for temporal and multi-spectral satellite imagery. *Advances in Neural Information Processing Systems*, 35:197–211, 2022. 1, 6

[8] Jiequan Cui, Zhisheng Zhong, Shu Liu, Bei Yu, and Jiaya Jia. Parametric contrastive learning. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 715–724, 2021. 3

[9] Jian Ding, Nan Xue, Gui-Song Xia, Xiang Bai, Wen Yang, Michael Yang, Serge Belongie, Jiebo Luo, Mihai Datcu, Marcello Pelillo, and Liangpei Zhang. Object detection in aerial images: A large-scale benchmark and challenges. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2021. 6

[10] Bowen Dong, Pan Zhou, Shuicheng Yan, and Wangmeng Zuo. Lpt: Long-tailed prompt tuning for image classification. In *The Eleventh International Conference on Learning Representations*, 2022. 3

[11] Stefan Falkner, Aaron Klein, and Frank Hutter. Bohb: Robust and efficient hyperparameter optimization at scale. In *International conference on machine learning*, pages 1437–1446. PMLR, 2018. 3

[12] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017. 3

[13] Chelsea Finn, Aravind Rajeswaran, Sham Kakade, and Sergey Levine. Online meta-learning. In *International conference on machine learning*, pages 1920–1930. PMLR, 2019. 3

[14] Africa Ixmuca Flores-Anderson, Kelsey E Herndon, Rajesh Bahadur Thapa, and Emil Cherrington. The sar handbook: comprehensive methodologies for forest monitoring and biomass estimation. Technical report, NASA SERVIR Global Program, 2019. 1

[15] Zihao Fu, Haoran Yang, Anthony Man-Cho So, Wai Lam, Lidong Bing, and Nigel Collier. On the effectiveness of parameter-efficient fine-tuning. In *Proceedings of the AAAI conference on artificial intelligence*, pages 12799–12807, 2023. 3

[16] P Goyal. Accurate, large minibatch sg d: training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017. 6

[17] Demi Guo, Alexander M Rush, and Yoon Kim. Parameter-efficient transfer learning with diff pruning. *arXiv preprint arXiv:2012.07463*, 2020. 3

[18] Xin Guo, Jiangwei Lao, Bo Dang, Yingying Zhang, Lei Yu, Lixiang Ru, Liheng Zhong, Ziyuan Huang, Kang Wu, Dingxiang Hu, et al. Skysense: A multi-modal remote sensing foundation model towards universal interpretation for earth observation imagery. *arXiv preprint arXiv:2312.10115*, 2023. 1

[19] Liu Hanxiao, Simonyan Karen, and Yang Yiming. Darts: Differentiable architecture search. In *International Conference on Learning Representations*, 2019. 3

[20] Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. Towards a unified view of parameter-efficient transfer learning. In *International Conference on Learning Representations*, 2022. 2, 3

[21] Danfeng Hong, Bing Zhang, Xuyang Li, Yuxuan Li, Chenyu Li, Jing Yao, Naoto Yokoya, Hao Li, Pedram Ghamisi, Xiuping Jia, et al. Spectralgpt: Spectral remote sensing foundation model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024. 1

[22] Xiyue Hou, Wei Ao, Qian Song, Jian Lai, Haipeng Wang, and Feng Xu. Fusar-ship: Building a high-resolution sar-ais matchup dataset of gaofen-3 for ship detection and recognition. *Science China Information Sciences*, 63:1–19, 2020. 6

[23] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International conference on machine learning*, pages 2790–2799. PMLR, 2019. 1, 3

[24] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International conference on machine learning*, pages 2790–2799. PMLR, 2019. 2, 3

[25] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021. 1, 2, 3, 4

[26] Max Jaderberg, Valentin Dalibard, Simon Osindero, Wojciech M Czarnecki, Jeff Donahue, Ali Razavi, Oriol Vinyals, Tim Green, Iain Dunning, Karen Simonyan, et al. Population based training of neural networks. *arXiv preprint arXiv:1711.09846*, 2017. 3

[27] Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge Belongie, Bharath Hariharan, and Ser-Nam Lim. Visual prompt tuning. In *European Conference on Computer Vision*, pages 709–727. Springer, 2022. 2

[28] Ding Jian, Xue Nan, Long Yang, Xia Gui-Song, and Qikai Lu. Learning roi transformer for detecting oriented objects in aerial images. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 6

[29] Bingyi Kang, Saining Xie, Marcus Rohrbach, Zhicheng Yan, Albert Gordo, Jiashi Feng, and Yannis Kalantidis. Decoupling representation and classifier for long-tailed recognition. *arXiv preprint arXiv:1910.09217*, 2019. 2, 3

[30] Jaehong Kim, Sangyeul Lee, Sungwan Kim, Moonsu Cha, Jung Kwon Lee, Youngduck Choi, Yongseok Choi, Dong-Yeon Cho, and Jiwon Kim. Auto-meta: Automated gradient based meta learner search. *arXiv preprint arXiv:1806.06927*, 2018. 3

[31] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. *book*, 2009. 5

[32] Darius Lam, Richard Kuzma, Kevin McGee, Samuel Dooley, Michael Laielli, Matthew Klaric, Yaroslav Bulatov, and Brendan McCord. xview: Objects in context in overhead imagery. *arXiv preprint arXiv:1802.07856*, 2018. 1

[33] Neal Lawton, Anoop Kumar, Govind Thattai, Aram Galstyan, and Greg Ver Steeg. Neural architecture search for parameter-efficient fine-tuning of large pre-trained language models. *arXiv preprint arXiv:2305.16597*, 2023. 3

[34] Songlin Lei, Dongdong Lu, Xiaolan Qiu, and Chibiao Ding. Srsdd-v1. 0: A high-resolution sar rotation ship detection dataset. *Remote Sensing*, 13(24):5104, 2021. 6

[35] Zhenguo Li, Fengwei Zhou, Fei Chen, and Hang Li. Meta-sgd: Learning to learn quickly for few-shot learning. *arXiv preprint arXiv:1707.09835*, 2017. 3

[36] 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. 3

[37] 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.

[38] Yaoyao Liu, Yingying Li, Bernt Schiele, and Qianru Sun. Wakening past concepts without past data: Class-incremental learning from online placebos. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2226–2235, 2024. 3

[39] Ziwei Liu, Zhongqi Miao, Xiaohang Zhan, Jiayun Wang, Boqing Gong, and Stella X Yu. Large-scale long-tailed recognition in an open world. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2537–2546, 2019. 5

[40] Lei Ma, Yu Liu, Xueliang Zhang, Yuanxin Ye, Gaofei Yin, and Brian Alan Johnson. Deep learning in remote sensing applications: A meta-analysis and review. *ISPRS journal of photogrammetry and remote sensing*, 152:166–177, 2019. 1

[41] Teli Ma, Shijie Geng, Mengmeng Wang, Jing Shao, Jiasen Lu, Hongsheng Li, Peng Gao, and Yu Qiao. A simple long-tailed recognition baseline via vision-language model. *arXiv preprint arXiv:2111.14745*, 2021. 3

[42] Aditya Krishna Menon, Sadeep Jayasumana, Ankit Singh Rawat, Himanshu Jain, Andreas Veit, and Sanjiv Kumar. Long-tail learning via logit adjustment. *arXiv preprint arXiv:2007.07314*, 2020. 5, 6

[43] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on Machine Learning*, pages 8748–8763. PMLR, 2021. 1, 6

[44] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022. 1

[45] T-YLPG Ross and GKHP Dollár. Focal loss for dense object detection. In *proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2980–2988, 2017. 3

[46] Gary A Shaw and Hsiaohua K Burke. Spectral imaging for remote sensing. *Lincoln laboratory journal*, 14(1):3–28, 2003. 1

[47] 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. 1, 3, 6

[48] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 25, 2012. 3

[49] Changyao Tian, Wenhai Wang, Xizhou Zhu, Jifeng Dai, and Yu Qiao. Vl-ltr: Learning class-wise visual-linguistic representation for long-tailed visual recognition. In *European conference on computer vision*, pages 73–91. Springer, 2022. 3

[50] Zichen Tian, Zhaozheng Chen, and Qianru Sun. Learning de-biased representations for remote-sensing imagery. In *Advances in Neural Information Processing Systems*, 2024. 1, 6

[51] Grant Van Horn, Oisin Mac Aodha, Yang Song, Yin Cui, Chen Sun, Alex Shepard, Hartwig Adam, Pietro Perona, and Serge Belongie. The inaturalist species classification and detection dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8769–8778, 2018. 5

[52] Yi Wang, Conrad M Albrecht, Nassim Ait Ali Braham, Lichao Mou, and Xiao Xiang Zhu. Self-supervised learning in remote sensing: A review. *IEEE Geoscience and Remote Sensing Magazine*, 10(4):213–247, 2022. 1

[53] Gui-Song Xia, Xiang Bai, Jian Ding, Zhen Zhu, Serge Belongie, Jiebo Luo, Mihai Datcu, Marcello Pelillo, and Liangpei Zhang. Dota: A large-scale dataset for object detection in aerial images. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 6

[54] Gui-Song Xia, Xiang Bai, Jian Ding, Zhen Zhu, Serge Belongie, Jiebo Luo, Mihai Datcu, Marcello Pelillo, and Liangpei Zhang. Dota: A large-scale dataset for object detection in aerial images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3974–3983, 2018. 1

[55] Yi Xin, Siqi Luo, Haodi Zhou, Junlong Du, Xiaohong Liu, Yue Fan, Qing Li, and Yuntao Du. Parameter-efficient fine-tuning for pre-trained vision models: A survey. *arXiv preprint arXiv:2402.02242*, 2024. 1

[56] Yuzhe Yang and Zhi Xu. Rethinking the value of labels for improving class-imbalanced learning. *Advances in neural information processing systems*, 33:19290–19301, 2020. 2, 3

[57] Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. 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)*, pages 1–9, 2022. 1, 3

[58] Jingyi Zhang, Jiaxing Huang, Sheng Jin, and Shijian Lu. Vision-language models for vision tasks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024. 1

[59] Yifan Zhang, Bingyi Kang, Bryan Hooi, Shuicheng Yan, and Jiashi Feng. Deep long-tailed learning: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023. 3

[60] Zhengning Zhang, Lin Zhang, Yue Wang, Pengming Feng, and Ran He. Shiprsimagenet: A large-scale fine-grained dataset for ship detection in high-resolution optical remote sensing images. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 14:8458–8472, 2021. 1

[61] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE transactions on pattern analysis and machine intelligence*, 40(6):1452–1464, 2017. 5

[62] Boyan Zhou, Quan Cui, Xiu-Shen Wei, and Zhao-Min Chen. 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*, pages 9719–9728, 2020. 3

[63] Xiao Xiang Zhu, Devis Tuia, Lichao Mou, Gui-Song Xia, Liangpei Zhang, Feng Xu, and Friedrich Fraundorfer. Deep learning in remote sensing: A comprehensive review and list of resources. *IEEE geoscience and remote sensing magazine*, 5(4):8–36, 2017. 1

[64] Xiao Xiang Zhu, Sina Montazeri, Mohsin Ali, Yuansheng Hua, Yuanyuan Wang, Lichao Mou, Yilei Shi, Feng Xu, and Richard Bamler. Deep learning meets sar: Concepts, models, pitfalls, and perspectives. *IEEE Geoscience and Remote Sensing Magazine*, 9(4):143–172, 2021. 1

[65] B Zoph. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016. 3