TR-PTS: Task-Relevant Parameter and Token Selection for Efficient Tuning

Siqi Luo[1*], Haoran Yang[1*], Yi Xin[2*], Mingyang Yi[3], Guangyang Wu[1],
Guangtao Zhai[1], Xiaohong Liu[1,4†]

[1]Shanghai Jiao Tong University [2]Shanghai Innovation Institute [3]Renmin University of China
[4]Suzhou Key Laboratory of Artificial Intelligence

{siqiluo647, yhr-73, wu.guang.young, zhaiguangtao, xiaohongliu}@sjtu.edu.cn
xinyi@smail.nju.edu.cn yimingyang@ruc.edu.cn

Abstract

*Large pre-trained models achieve remarkable performance in vision tasks but are impractical for fine-tuning due to high computational and storage costs. Parameter-Efficient Fine-Tuning (PEFT) methods mitigate this issue by updating only a subset of parameters; however, most existing approaches are task-agnostic, failing to fully exploit task-specific adaptations, which leads to suboptimal efficiency and performance. To address this limitation, we propose **T**ask-**R**elevant **P**arameter and **T**oken **S**election (TR-PTS), a task-driven framework that enhances both computational efficiency and accuracy. Specifically, we introduce **Task-Relevant Parameter Selection**, which utilizes the Fisher Information Matrix (FIM) to identify and fine-tune only the most informative parameters in a layer-wise manner, while keeping the remaining parameters frozen. Simultaneously, **Task-Relevant Token Selection** dynamically preserves the most informative tokens and merges redundant ones, reducing computational overhead. By jointly optimizing parameters and tokens, TR-PTS enables the model to concentrate on task-discriminative information. We evaluate TR-PTS on benchmark, including FGVC and VTAB-1k, where it achieves state-of-the-art performance, surpassing full fine-tuning by 3.40% and 10.35%, respectively. The code are available at https://github.com/synbol/TR-PTS.*

1. Introduction

Pre-trained on large-scale datasets and subsequently fine-tuned for downstream tasks, this paradigm has become the standard framework across various domains, including Natural Language Processing (NLP) [2, 5], Computer Vision (CV) [8, 20, 24, 33, 35], and others [19, 32]. Traditionally, fine-tuning a pre-trained model to a specific task involves adjusting all parameters, a method known as full fine-tuning.

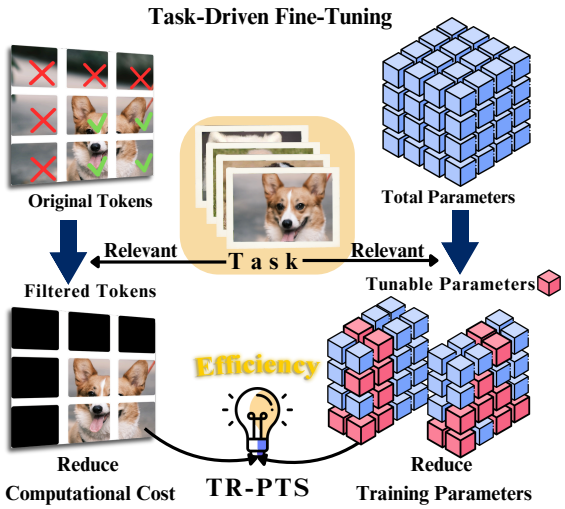


Figure 1. **Overview of Our TR-PTS**. TR-PTS selectively retains task-relevant tokens and parameters for efficient fine-tuning.

However, with state-of-the-art models now comprising billions or even trillions of parameters [22, 27], this conventional approach has become increasingly impractical due to its exorbitant computational and storage requirements.

Mitigating the inefficiencies of full fine-tuning has led researchers in the CV domain to explore Parameter-Efficient Fine-Tuning (PEFT) techniques [18, 31, 36, 39], which update only a subset of model parameters or introduce lightweight adaptation modules [11, 13, 14]. However, as summarized in Table 1, existing methods still face several challenges: **(1) Inference Overhead:** Some methods, such as VPT [14], introduce additional learnable modules that increase computational cost during inference. Others, such as LoRA [13], use reparameterization techniques that can be merged into the backbone at inference time and therefore avoid extra cost. **(2) Limited Task-Aware Adaptation:** While some methods have begun to incorporate task-specific adaptation, such as GPS [39], many still use uniform tuning strategies that overlook the varying importance of model components across tasks. **(3) Disjoint Optimization:** Most

*Equal Contribution.
†Corresponding Author.

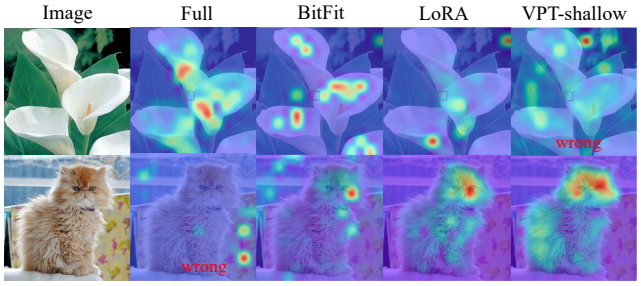


Figure 2. **Visualizing the Impact of Each Token on the Final Prediction.** The label "wrong" indicates an incorrect classification. These results demonstrate that the final prediction relies primarily on a subset of the most task-relevant tokens.

approaches optimize parameter selection and token processing separately, although the informativeness of tokens depends heavily on the task, particularly in ViT-based models. As shown in Figure 2, token importance varies across tasks and should be considered during optimization. This motivates the development of a unified solution that considers both task-relevant parameters and informative tokens.

To address these challenges, we propose **T**ask-**R**elevant **P**arameter and **T**oken **S**election (TR-PTS), a novel framework that unifies task-driven parameter selection and token refinement. As illustrated in Figure 1, when a task arrives, the model's adaptation is manifested both in parameter sensitivity and token informativeness. Guided by this principle, our method dynamically identifies the most relevant parameters and tokens for the task at hand, enabling the model to concentrate computation on the most discriminative information while avoiding redundant updates.

- **Task-Relevant Parameter Selection:** Rather than relying on gradient magnitudes [39], which can be affected by optimization noise and may not accurately reflect task importance, we employ the Fisher Information Matrix (FIM) [16] to quantify parameter sensitivity. We then select the most critical parameters and use a layer-wise allocation strategy to distribute trainable connections according to each layer's task relevance, reducing the number of trainable parameters while preserving adaptability.
- **Task-Relevant Token Selection:** We leverage attention scores from the $[CLS]$ token to retain the most informative tokens while merging the rest through weighted averaging. This allows the model to concentrate attention on discriminative content and reduces computational overhead without sacrificing accuracy.

Empirical observations during system design revealed a notable correlation between parameter sparsity and token redundancy: layers with fewer task-relevant parameters tend to encode less informative tokens. Leveraging this insight, we formulate a coordinated selection strategy that applies token reduction preferentially to parameter-sparse layers. This unified, task-aware mechanism enhances computational efficiency while preserving representational integrity, thereby achieving a more favorable trade-off between accuracy and resource consumption.

Method	Task Params. Adaptive	Task Tokens Adaptive	Memory Efficient	Train/Inference Efficient
Full [14]	✗	✗	✗	✗
Linear [14]	✗	✗	✓	✓
BitFit [36]	✓	✓	✗	✗
Adapter [11]	✗	✗	✗	✗
VPT [14]	✗	✗	✗	✗
LoRA [12]	✗	✗	✗	✗
SSF [17]	✗	✗	✗	✗
GPS [39]	✓	✗	✗	✗
TR-PTS (Ours)	✓	✓	✓	✓

Table 1. **Comparison of Different Fine-Tuning Methods.** Our method adaptively selects task-relevant parameters and tokens, allowing it to be applied across various model architectures without adding extra parameters during either training or inference.

We evaluate TR-PTS across a diverse set of 24 image recognition tasks. Our approach demonstrates state-of-the-art performance compared to other PEFT methods while achieving an optimal balance between performance, trainable parameters, and computational cost, as illustrated in Figure 1. Compared to full fine-tuning, TR-PTS achieves 3.40% (FGVC) and 10.35% (VTAB) improvement of the accuracy while tuning only 0.60% and 0.34% parameters of the pre-trained model. Additionally, the time required for both fine-tuning and the inference process is significantly reduced, as detailed in the referenced Section 4.2.

2. Related Work

2.1. Visual Efficient Fine-Tuning

In the survey on visual efficient fine-tuning [34], existing methods are generally categorized into five primary approaches: 1) *Adapter Tuning* methods [3, 11] introduce small-scale neural modules (adapters) into Transformer layers, tuning only these adapters for model adaptation; 2) *Prompt Tuning* methods [14] enhance the original input by adding additional visual prompts; 3) *Specification Tuning* [36, 39] directly modifies a specific subset of parameters in the Transformer to improve efficiency; 4) *Reparameterization Tuning* methods [12, 17] introduce new learnable parameters during training, which are later integrated into the original Transformer layers through reparameterization during inference; 5) *Unified-based Tuning* methods [38] provide a unified framework that integrates various fine-tuning techniques into a single, cohesive architecture.

The method we propose differs fundamentally from these existing approaches, as shown in Table 1. While most of the aforementioned methods focus primarily on the structural design of the model, our approach emphasizes the identification and tuning of important parameters and tokens that are specifically relevant to the downstream tasks.

2.2. Token Reduction for Efficient ViT

Since the introduction of ViTs, researchers have explored ways to enhance their efficiency by reducing redundant to-

kens. Existing methods primarily fall into two categories: token pruning [9, 30] and token merging [1, 25]. Pruning-based approaches remove less important tokens to lower computational complexity and merging-based methods preserve more information by fusing similar tokens, reducing sequence length. The method we propose combines the advantages of pruning and merging. We dynamically select task-relevant important tokens for retention. Instead of discarding unimportant tokens, we perform weighted merging based on task relevance.

3. Method

Our method is built upon Vision Transformer (ViT) [7] architecture. In the following, we first review the ViT architecture in Section 3.1. Subsequently, we present the proposed **T**ask-**R**elevant **P**arameter and **T**oken **S**election (TR-PTS) in Section 3.2 and Section 3.3. Finally, we present the integration of these components into a unified Task-Driven Fine-Tuning strategy in Section 3.4.

3.1. Preliminary

The standard ViT consists of a patch embedding layer and a stack of Transformer layers. Given an image $I \in \mathbb{R}^{H\times W\times C}$, the patch embedding layer first splits and flatten the image I into sequential patches $I_p \in \mathbb{R}^{N\times(P^2C)}$, where (H, W) represents the height and width of the input image, (P, P) is the resolution of each image patch, C denotes the number of channels, and $N = HW/P^2$ is the number of image tokens. Then, the patches I_p are mapped to $X_0 = [x_1, x_2, ..., x_N] \in \mathbb{R}^{N\times d}$ using a trainable linear projection. The inputs to the Transformer layers consist of X_0 along with a prepended $[CLS]$ token.

3.2. Task-Relevant Parameter Selection

Existing gradient-based parameter selection methods [39] often fail to capture task relevance effectively, leading to uniform parameter selection across layers. To address this, we introduce a layer-wise selection strategy leveraging the Fisher Information Matrix (FIM) [16], which quantifies each parameter's contribution to task-specific adaptations.

3.2.1. Fisher Information Matrix (FIM)

FIM evaluates the sensitivity of model outputs to perturbations in parameters. Higher FIM values indicate stronger task relevance, suggesting priority for fine-tuning. Formally, the FIM for model parameters θ is:

$$\mathcal{F}(\theta) = \mathbb{E}\left[\left(\frac{\partial \log p(y|x;\theta)}{\partial\theta}\right)\left(\frac{\partial \log p(y|x;\theta)}{\partial\theta}\right)^{\top}\right], \quad (1)$$

where $p(y|x;\theta)$ represents the conditional probability of output y given input x. This matrix characterizes the covariance of the gradient of the log-likelihood with respect to the model parameters, thereby identifying the parameters that exert the most substantial influence on task performance.

In practice, we approximate the FIM using the gradients of the cross-entropy loss function $\mathcal{L}_{\text{CE}}$ in image classification tasks, as its gradient aligns with the first-order derivatives of the log-likelihood function. Specifically, we use the squared gradients to approximate the diagonal of the FIM:

$$\mathcal{F}(\theta) \approx \mathbb{E}_{(x,y)\sim D}\left[\left(\frac{\partial \mathcal{L}_{\text{CE}}}{\partial\theta}\right)^2\right]. \quad (2)$$

This approximation avoids the complexity of explicitly computing second-order derivatives while providing an efficient estimate of parameter importance in relation to the target task.

3.2.2. Layer-Wise Parameter Allocation

Task-Relevant Layer Importance. Instead of selecting parameters uniformly across layers, we introduce a layer-aware allocation strategy to ensure balanced adaptation. We compute task-driven layer-wise importance scores using the Fisher Information Matrix (FIM), allowing us to allocate trainable parameters based on their task relevance.

Once the FIM is computed for all parameters, we first select the top $M\%$ of parameters with the highest FIM values. Let $I_{\text{top-M}}$ denote this subset of selected parameters. The contribution of each layer l is computed as:

$$w_l = \frac{|I_{\text{top-M}} \cap L_l|}{|I_{\text{top-M}}|}, \quad (3)$$

where L_l represents the parameters in layer l. The computed w_l values determine the relative importance of each layer, ensuring that task-relevant layers receive more trainable parameters.

Adaptive Layer-Wise Parameter Selection. Inspired by the Gradient-based Parameter Selection (GPS) method [39], we introduce a selection strategy that ensures each neuron in layer L fine-tunes exactly C_l connections. To dynamically allocate trainable connections per layer, we normalize w_l relative to the least important layer:

$$C_l = \max\left(1, \frac{w_l}{\min(w)} \cdot C_{\text{min}}\right), \quad (4)$$

where C_{min} represents the minimum number of selected connections per neuron in the least important layer, and $\min(w)$ denotes the smallest observed importance ratio. This formulation guarantees that every layer retains at least one active connection while allocating a greater number of parameters to task-critical layers.

Within each layer l, we further refine the selection by identifying a subset of task-relevant connections per neuron based on their Fisher Information Matrix (FIM) scores:

$$S_l = \{\theta_i \mid i \in \arg\text{top-}C_l\left(\mathcal{F}(\theta_i)\right), \theta_i \in \mathcal{N}_l\}, \quad (5)$$

where $\mathcal{N}_l$ denotes the set of all input connections to neurons in layer l. This selection strategy ensures that each neuron retains a subset of high-FIM parameters, preventing any part of the network from becoming entirely inactive.

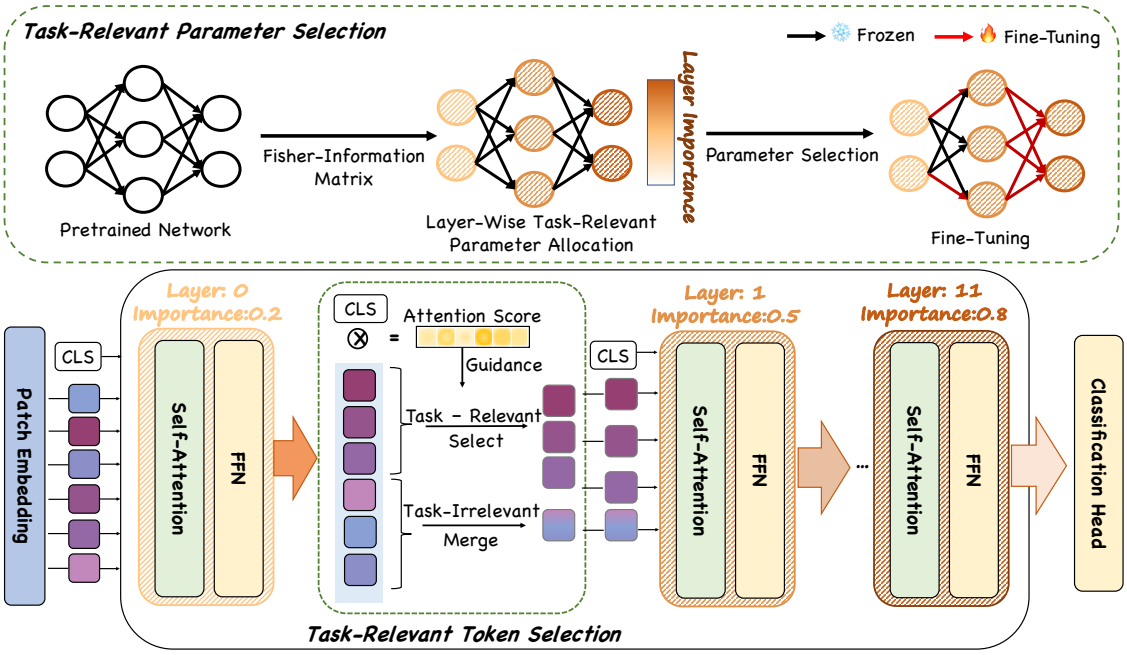


Figure 3. **Architecture of Our Proposed TR-PTS Framework. Top:** For Task-Relevant Parameter Selection, we use the Fisher Information Matrix to determine each layer's importance and dynamically set the number of trainable connections per neuron (N_l); critical layers receive more parameters, while every layer updates at least one connection. **Bottom:** For Task-Relevant Token Selection, attention scores from the $[CLS]$ token identify key tokens, and less informative tokens are merged via weighted averaging. Trainable parameters are allocated per layer in proportion to their Fisher scores.

Final Task-Relevant Parameter Set. Given the neuron-level selection S_n and the dynamically determined per-layer C_l, the final task-relevant parameter set is:

$$\Theta_{\mathcal{T}} = \bigcup_l S_l, \quad \text{where } |S_l| = C_l \cdot |\mathcal{N}_l|, \tag{6}$$

Here, $|\cdot|$ denotes the cardinality of a set, i.e., the number of elements it contains. By focusing on task-relevant parameters at both the layer and neuron levels, our method minimizes redundant computations while ensuring fine-tuning efficiency.

3.3. Task-Relevant Token Selection

In Vision Transformers (ViTs), images are split into patch tokens processed by self-attention, where token importance is inherently influenced by parameter distribution. Our Task-Relevant Parameter and Token Selection (TR-PTS) framework optimizes attention by selecting the most informative parameters, which in turn adaptively guides token selection.

To further enhance computational efficiency, we perform Token Selection and Token Merge progressively. Specifically, token selection is strategically applied to layers with sparse task-relevant parameters to ensure a balanced trade-off between parameter fine-tuning and token efficiency. At each refining layer, we retain only the most task-relevant tokens while merging the less informative ones into a single token. This approach minimizes redundant computation and allows the model to focus on the most crucial tokens, potentially improving overall performance.

3.3.1. Token Selection

In Vision Transformers (ViTs), the $[CLS]$ token acts as a global feature aggregator, attending to all image tokens across layers to accumulate task-discriminative information for classification.

In self-attention, tokens contribute differently to the final output, with attention scores reflecting their importance. Tokens with higher scores influence classification more, while lower-weighted ones carry less discriminative information.

Formally, in a Transformer layer, the self-attention mechanism computes attention scores using the scaled dot-product formula:

$$A = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right), \tag{7}$$

where Q and K are the query and key matrices of all tokens, and d is the key dimension. The attention score a_i of an image token x_i relative to the $[CLS]$ token is given by:

$$a_i = \frac{\exp(q_{\text{CLS}} \cdot k_i)}{\sum_{j=1}^{N} \exp(q_{\text{CLS}} \cdot k_j)}, \tag{8}$$

where q_{CLS} is the query vector of the $[CLS]$ token, and k_i is the key vector of token x_i. This score quantifies how much information token x_i contributes to the final $[CLS]$ representation, making it a reliable indicator of task relevance.

To retain the most informative tokens, we introduce a select rate $\rho \in (0, 1]$, which determines the fraction of tokens to be preserved. Specifically, we retain the top $\lfloor \rho N \rfloor$ tokens based on their attention scores:

$$X_{\text{selected}} = \{x_i \mid i \in \arg\,\text{top-}\lfloor \rho N \rfloor (a)\}\,. \tag{9}$$

By adaptively selecting tokens based on the attention distribution rather than a fixed number, our approach ensures that token selection remains flexible across different layers and tasks. This not only reduces redundant computation but also maintains high task performance.

3.3.2. Token Merge

While Token Selection removes redundant tokens, Token Merge ensures that information from discarded tokens is preserved rather than entirely ignored. Instead of discarding low-attention tokens, we merge them into a single aggregated token using a weighted averaging process.

Let $\mathcal{I}$ be the set of less informative tokens, i.e., tokens that were not selected based on attention scores. We compute a fused token x_{merged} as:

$$x_{\text{merged}} = \frac{\sum_{i \in \mathcal{I}} a_i x_i}{\sum_{i \in \mathcal{I}} a_i}. \tag{10}$$

At each refining layer, token selection and merging are performed to progressively refine the token sequence. This merged token is then appended to the selected tokens to form the refined token sequence:

$$X_{\text{refined}} = \{x_{\text{CLS}}, X_{\text{selected}}, x_{\text{merged}}\}. \tag{11}$$

By merging the discarded tokens into a single representation, we retain global information while significantly reducing the number of tokens processed in subsequent layers. This progressive refinement reduces the token sequence length as the model deepens, lowering both computation and memory usage.

3.4. Task-Driven Fine-Tuning

After identifying task-relevant parameters (see Section 3.2) and task-relevant tokens (see Section 3.3), we integrate both components into a cooperative fine-tuning framework, where token selection and parameter optimization interactively reinforce each other to enhance task adaptation. Let Θ denote the full set of pre-trained model parameters, and $\Theta_{\mathcal{T}}$ represent the subset identified as critical for the target task. To ensure that only the most relevant parameters are updated, we introduce a binary mask M over Θ:

$$M_i = \begin{cases} 1, & \text{if } \theta_i \in \Theta_{\mathcal{T}}, \\ 0, & \text{otherwise.} \end{cases} \tag{12}$$

The model is trained by minimizing the loss function $\mathcal{L}$, computed based on the refined token representation:

$$\min_{\Theta} \mathcal{L}\big(f(X_{\text{ref}}; \Theta),\, y\big), \tag{13}$$

while ensuring that only the parameters in $\Theta_{\mathcal{T}}$ are updated.

The update rule at iteration t is formulated as:

$$\Theta^{(t+1)} = \Theta^{(t)} - \eta \left(M \odot \nabla_{\Theta} \mathcal{L}\big(f(X_{\text{ref}}; \Theta^{(t)}),\, y\big) \right), \tag{14}$$

where η is the learning rate and $\odot$ denotes element-wise multiplication. This ensures that gradients are only applied to the task-relevant parameters while the remaining parameters stay frozen.

By jointly optimizing parameter selection and token refinement, our approach enables a mutual enhancement process, where the refined tokens guide more task-specific parameter updates, and the optimized parameters, in turn, refine the token representations. This co-adaptive learning strategy significantly reduces redundant computations and memory usage, ensuring that ViTs focus on the most task-discriminative features for efficient adaptation.

4. Experiments

4.1. Experimental settings

Datasets. We conduct our experiments primarily on a series of datasets categorized into three types as detailed below: 1) ***FGVC***: Fine-Grained Visual Classification (FGVC) benchmark [14] includes five downstream tasks, which are CUB-200-2011 [29], NABirds [28], Oxford Flowers [21], Stanford Dogs [4] and Stanford Cars [10]. 2) ***VTAB-1k***: Visual Task Adaptation Benchmark (VTAB) [37] contains of 19 visual classification tasks, which are grouped into three sets: Natural, Specialized, and Structured. Each task in VTAB-1k contains 1000 training example.

Pre-Trained Backbones. To ensure a fair comparison, we align with most of efficient fine-tuning methods [14, 17, 39] and adopt the plain Vision Transformer [6], *i.e.*, ViT-Base (ViT-B/16) as our backbone model and pre-train the model with both supervised method. Specifically, we directly use the ImageNet-21k [26] supervised pre-trained model.

Baseline Methods. We compare our TR-PTS with a variety of fine-tuning protocols that can be mainly categorized into three types [34]: 1) ***Full Fine-tuning***: the most commonly used protocol updating all parameters of the whole model during tuning. 2) ***Partial-based Tuning***: This kind of method concentrates on updating only a small subset of inherent parameters while maintaining the majority of the model's parameters unchanged during the adaptation process, including Linear Probing, BitFit [36], LoRA [12], SSF [17], and GPS [39]. 3) ***Addition-based Tuning***: This kind of method involves incorporating additional trainable modules or parameters into pre-trained backbones to learn task-specific information, including Adapter [11], AdaptFormer [3], VPT [14].

Implementation Details. We conduct our experiments on two main benchmarks: FGVC and VTAB-1k. For a fair comparison with existing PEFT methods, models are fine-tuned using the Adam optimizer [15] with a cosine learning rate decay schedule for 100 epochs. All experiments are implemented using the PyTorch framework [23].

Method \ Dataset	Natural							Specialized				Structured									
	CIFAR-100	Caltech101	DTD	Flowers102	Pets	SVHN	Sun397	Patch Camelyon	EuroSAT	Resisc45	Retinopathy	Clevr/count	Clevr/distance	DMLab	KITTI/distance	dSprites/loc	dSprites/ori	SmallNORB/azi	SmallNORB/ele	Mean Acc.	Params. (%)
Full [14]	68.9	87.7	64.3	97.2	86.9	87.4	38.8	79.7	95.7	84.2	73.9	56.3	58.6	41.7	65.5	57.5	46.7	25.7	29.1	65.57	100.00
Linear [14]	63.4	85.0	64.3	97.0	86.3	36.6	51.0	78.5	87.5	68.6	74.0	34.3	30.6	33.2	55.4	12.5	20.0	9.6	19.2	53.00	0.05
BitFit [36]	72.8	87.0	59.2	97.5	85.3	59.9	51.4	78.7	91.6	72.9	69.8	61.5	55.6	32.4	55.9	66.6	40.0	15.7	25.1	62.05	0.16
Adapter [11]	74.1	86.1	63.2	97.7	87.0	34.6	50.8	76.3	88.0	73.1	70.5	45.7	37.4	31.2	53.2	30.3	25.4	13.8	22.1	55.82	0.31
AdaptFormer [3]	70.8	91.2	70.5	99.1	90.9	86.6	54.8	83.0	95.8	84.4	76.3	81.9	64.3	49.3	80.3	76.3	45.7	31.7	41.1	72.32	0.24
LoRA [12]	68.1	91.4	69.8	99.0	90.5	86.4	53.1	85.1	95.8	84.7	74.2	83.0	**66.9**	50.4	81.4	80.2	46.6	32.2	41.1	72.63	0.90
VPT-Shallow [14]	77.7	86.9	62.6	97.5	87.3	74.5	51.2	78.2	92.0	75.6	72.9	50.5	58.6	40.5	67.1	68.7	36.1	20.2	34.1	64.85	0.13
VPT-Deep [14]	78.8	90.8	65.8	98.0	88.3	78.1	49.6	81.8	96.1	83.4	68.4	68.5	60.0	46.5	72.8	73.6	47.9	**32.9**	37.8	69.43	0.70
SSF [17]	69.0	92.6	75.1	99.4	91.8	90.2	52.9	87.4	95.9	87.4	75.5	75.9	62.3	53.3	80.6	77.3	54.9	29.5	37.9	73.10	0.28
GPS [39]	81.1	**94.2**	**75.8**	99.4	91.7	**91.6**	52.4	87.9	**96.2**	86.5	76.5	79.9	62.6	**55.0**	82.4	84.0	55.4	29.7	**46.1**	75.18	0.25
TR-PTS (Ours)	**81.2**	93.9	75.1	**99.5**	**91.9**	91.0	**54.5**	**88.1**	95.7	**87.8**	**76.6**	**83.5**	63.2	54.8	**82.8**	**87.7**	**56.9**	31.8	**46.1**	**75.92**	0.34

Table 2. **Performance Comparisons on VTAB-1k with ViT-B/16 Models Pre-trained on ImageNet-21K.**

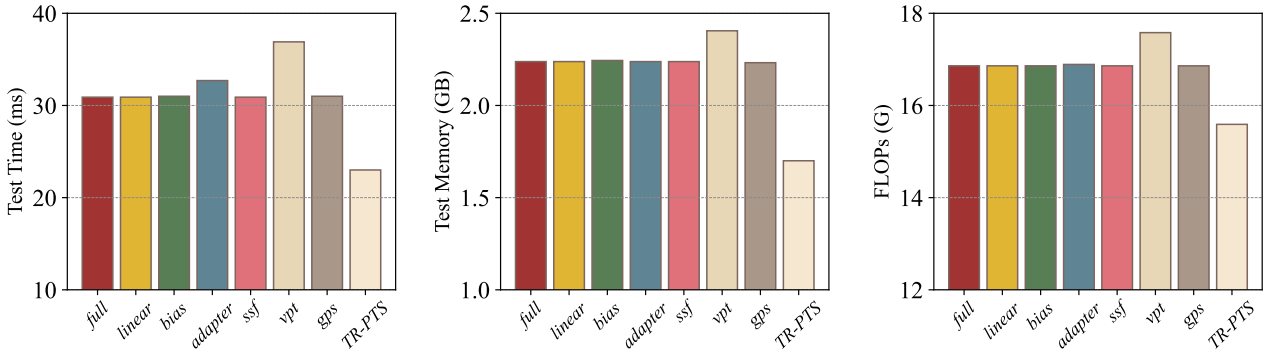


Figure 4. **Comparison of Different Methods in terms of Test Time (ms), Test Memory Usage (GB), and FLOPs (G).** The results show that the TR-PTS method achieves the lowest FLOPs and memory consumption while maintaining competitive test time.

Method \ Dataset	CUB-200 -2011	NABirds	Oxford Flowers	Stanford Dogs	Stanford Cars	Mean	Params.(%)
Full [14]	87.3	82.7	98.8	89.4	84.5	88.54	100.00
Linear [14]	85.3	75.9	97.9	86.2	51.3	79.32	0.21
BitFit [36]	88.4	84.2	98.8	91.2	79.4	88.40	0.33
Adapter [11]	87.1	84.3	98.5	89.8	68.6	85.66	0.48
AdaptFormer[3]	88.4	84.7	99.2	88.2	81.9	88.48	0.75
LoRA [12]	85.6	79.8	98.9	87.6	72.0	84.78	0.90
VPT-Shallow [14]	86.7	78.8	98.4	90.7	68.7	84.62	0.29
VPT-Deep [14]	88.5	84.2	99.0	90.2	83.6	89.11	0.99
SSF [17]	89.5	85.7	99.6	89.6	89.2	90.72	0.45
GPS [39]	89.9	86.7	**99.7**	92.2	90.4	91.78	0.77
TR-PTS (Ours)	**90.0**	**87.1**	99.6	**92.4**	**90.6**	**91.94**	0.60

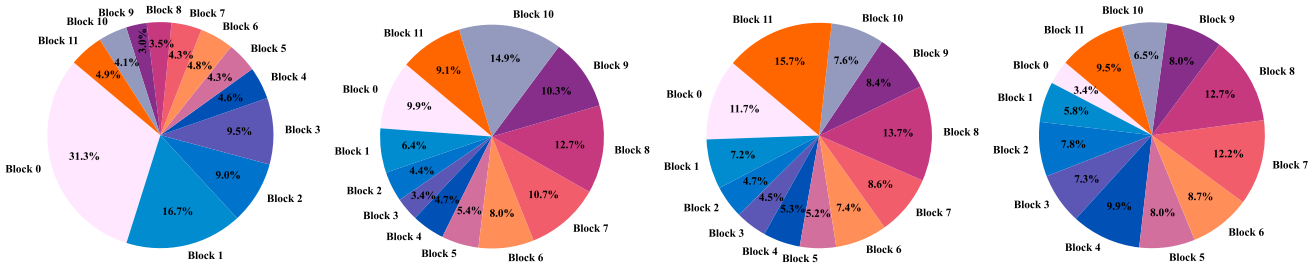
Table 3. **Performance Comparisons on Five FGVC Datasets with ViT-B/16 Models Pre-trained on ImageNet-21K.**

4.2. Main Properties and Analysis

We conduct a comprehensive evaluation on two benchmarks, VTAB and FGVC, which together consist of 24 diverse datasets. In our experiments, we compare our proposed approach with leading fine tuning protocols based on Top-1 accuracy and the percentage of fine-tuned parameters. In addition, we rigorously assess both computational and storage costs to demonstrate the efficiency of our approach, thereby validating its practical advantages in terms of both performance and resource utilization.

Comparisons on VTAB-1K. For the VTAB benchmark, our TR-PTS framework achieves substantial improvements over existing fine-tuning methods. As shown in Table 2, it attains an average top-1 accuracy of 75.92%, exceeding full fine-tuning by 10.35% and surpassing GPS by 0.74%, demonstrating superior performance on 13 subtasks and state-of-the-art results on 11. Notably, TR-PTS fine-tunes only 0.34% of model parameters, significantly reducing computational cost. The superior performance of TR-PTS stems from its task-relevant token and parameter selection strategies: layer-wise allocation refines only the most critical parameters, improving accuracy while minimizing parameter updates, whereas token selection enhances efficiency by prioritizing task-critical information. By selectively merging less informative tokens, our method prevents redundancy while preserving essential representations. These combined strategies enable TR-PTS to achieve both efficiency and robustness, making it highly effective for the VTAB benchmark.

Comparisons on FGVC. Table 3 shows that TR-PTS achieves the state of art performance on FGVC datasets, with an average accuracy of 91.94%, slightly surpassing GPS (91.78%) and and outperforming full fine-tuning by 3.40%. While the accuracy gains are marginal, fine-grained classification presents inherent challenges, with subtle inter-class differences limiting performance improvements. What distinguishes TR-PTS is its efficiency. Unlike other methods, our token selection strategy effectively reduces computational cost, as quantified by FLOPs (see Figure 4). These



(a) Sun397 (b) Flower102 (c) Dsprites/loc (d) Patch Camelyon

Figure 5. **Differernt Datasets Top 1% FIM Parameter Distribution.**

results demonstrate that TR-PTS not only delivers competitive accuracy in a demanding fine-grained classification setting but also significantly enhances computational efficiency.

Computational Cost. In Figure 4, we evaluate TR-PTS against various PEFT methods in terms of computational efficiency, with all experiments conducted on a single NVIDIA A100 GPU under consistent settings. Specifically, all methods are evaluated using a batch size of 32, input resolution of 224×224, and a ViT-B backbone. FLOPs are measured analytically, inference time is averaged over 500 forward passes, and memory usage is recorded after a single pass. TR-PTS achieves the shortest inference time, primarily due to its task-relevant token selection mechanism, which removes redundant tokens and accelerates processing. In terms of memory usage, TR-PTS consumes the least among all methods, unlike VPT, which introduces additional modules that increase runtime memory requirements. Regarding FLOPs, a key indicator of computational complexity, TR-PTS reports the lowest count by dynamically selecting and merging tokens, thereby reducing unnecessary operations. By comparison, GPS lacks structured token selection and incurs higher FLOPs. These results demonstrate that TR-PTS effectively enhances both token and parameter efficiency, reducing computational cost while preserving strong task performance.

4.3. Ablation Studies

Components Effectiveness. We analyze the contribution of each component in our framework: Task-Relevant Parameter Selection and Task-Relevant Token Selection. Table 4 presents the impact of integrating these components. To ensure a comprehensive evaluation, we select one task from each of the three major VTAB categories and use linear fine-tuning as the baseline. Introducing TR-PTS significantly improves performance, achieving gains of 75.2%, 2.5%, and 3.5% across datasets. Applying token selection at an optimal layer, without additional parameter selection or fine-tuning, yields accuracy improvements of 2.3%, 1.8%, and 0.2%, respectively. Conversely, parameter selection alone, without token selection, leads to increases of 72.6%, 2.4%, and 3.2%. The combination of both components in TR-PTS consistently enhances model performance. These results highlight the necessity of selecting both task-relevant tokens and parameters, demonstrating their complementary roles in optimizing fine-tuning for diverse downstream tasks.

TR-PTS		VTAB-1k		
Token Selection	Parameter Selecetion	dSprites/loc	Flower102	Sun397
✗	✗	12.5	97.0	51.0
✓	✗	14.8	98.8	51.2
✗	✓	85.1	99.4	54.2
✓	✓	**87.7**	**99.5**	**54.5**

Table 4. **Ablation Study on the impact of Task-Relevant Token Selection and Task-Relevant Parameter Selection.**

Analysis of Task-Relevant Parameter Selection. To assess the impact of our task-relevant parameter selection strategy, we analyze the layer-wise distribution of critical parameters and the overlap between task-specific parameter sets.

- **Layer-Wise Distribution of Critical Parameters:** The parameter distribution across datasets reflects task relevance, varying by dataset needs. As shown in Figure 5, Flower102 concentrates parameters in Blocks 8 and 10, relying more on high-level feature extraction, while lower layers contribute less. In contrast, Patch/Camelyon has a uniform distribution, indicating equal reliance on all blocks. Sun397, however, is dominated by Block 0, suggesting a greater dependency on low-level features. These results confirm that task-aware parameter selection is crucial, as different datasets prioritize different layers.
- **Overlap Between Task-Specific Parameter Sets:** As shown in Figure 7, similar tasks exhibit greater overlap, such as Resisc45 and Smallnorb/Azi (0.38), indicating shared task-relevant parameters. In contrast, Sun397 shows minimal overlap with Patch/Camelyon (0.17) and Dsprites/Loc (0.18), suggesting distinct parameter selection. Moderate overlap between Dsprites/Loc and Patch/Camelyon (0.28–0.32) implies partial parameter sharing while maintaining task specificity. Overall, the low parameter overlap validates the effectiveness of our adaptive selection strategy over uniform allocation.

Explore the Intrinsic Correlation Between Parameter Density and Token Redundancy. By analyzing token selection and parameter distribution, we observe a strong intrinsic correlation between token redundancy and parameter sparsity. In particular, layers with fewer task-relevant

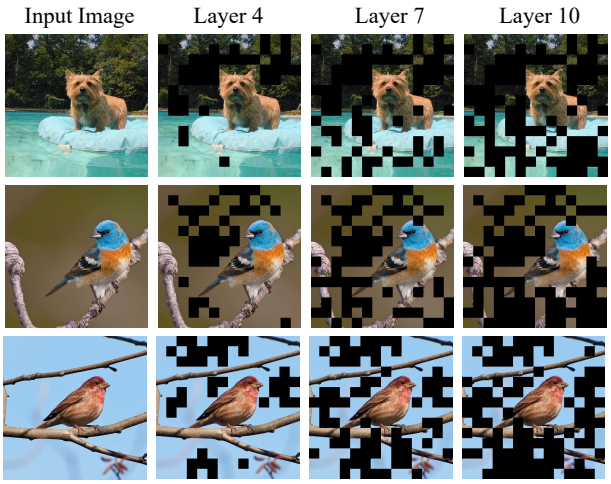


Figure 6. **Visualization of Task-Relevant Tokens Selected by TR-PTS Using ViT-B/16 with 12 Layers.** As the number of layers increases, our method increasingly focuses on task-relevant tokens.

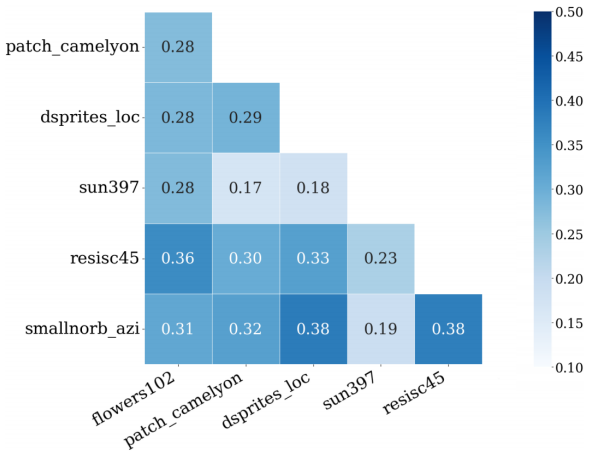


Figure 7. **Visualization of the Overlapping Rate Among Task-Driven Parameter Sets.** This heatmap shows the overlap between parameter sets across tasks. Darker shades indicate higher similarity, while lighter shades highlight task-specific differences.

parameters tend to encode less informative tokens, making them better candidates for token reduction. As shown in Table 5, applying token selection in layers with dense parameter updates, such as in Flower102 and Camelyon, often leads to performance drops. This indicates that pruning in highly task-relevant layers can disrupt essential task-specific information. Moreover, random token selection across layers results in inconsistent performance, further emphasizing the importance of a task-aware strategy. Motivated by this finding, we design a joint selection strategy that applies token reduction primarily in sparse parameter layers—those with fewer selected task-relevant connections. This "sparse insertion" method ensures that token selection minimizes interference with critical computation while still reducing redundancy. As a result, TR-PTS achieves a favorable balance between efficiency and accuracy, outperforming both dense and random strategies.

	Selection Ratio	Sun397	Flower102	Loc	Camelyon
Dense	*0.95*	53.5	99.3	85.2	87.3
	0.8	52.7	99.1	86.9	87.4
Random	*0.95*	54.0	99.3	85.9	87.9
	0.8	52.5	99.2	85.5	86.1
Sparse	*0.95*	**54.5**	**99.4**	**87.7**	**88.1**
	0.8	52.9	99.1	86.0	87.2

Table 5. **Comparison of Token Selection Positions.** "Dense" refers to layers rich in task-relevant parameters, where pruning may harm performance. "Sparse" applies selection in layers with fewer task-relevant parameters, making pruning more effective. "Random" selects tokens without a structured strategy.

Token Selection Visualization. To evaluate the effectiveness of our proposed Task-Relevant Token Selection strategy, we conduct a Token Selection Visualization experiment using ViT-B/16 with 12 layers. This experiment is performed on two fine-grained datasets, CUB-200-2011 and Stanford Dogs, with a token select rate of 0.8. We visualize the token selection process at different layers, specifically the 4th, 7th, and 10th layers, all within a single forward pass of the network. The results show that in the early layers (e.g., Layer 4), token retention is relatively widespread, capturing local features. In deeper layers (e.g., Layer 10), the model gradually focuses on foreground objects, such as birds in CUB-200-2011 and dogs in Stanford Dogs, effectively eliminating background noise. As the network deepens, more tokens are merged or refined, reducing computational redundancy while allowing the model to concentrate on the most critical image regions.

5. Conclusion

This work enhances the efficiency of adapting pre-trained Vision Transformers (ViT) during fine-tuning and inference. We propose Task-Relevant Parameter and Token Selection (TR-PTS), which improves efficiency from two perspectives. On the parameter side, it leverages the Fisher Information Matrix (FIM) to fine-tune only the most task-relevant parameters, reducing trainable parameters while preserving adaptation. On the token side, TR-PTS dynamically selects and merges tokens based on $[CLS]$ attention scores, retaining only the most informative ones to reduce redundancy and lower computational overhead. TR-PTS maintains strong task performance with reduced cost. Future work will extend it to segmentation, detection, and adaptive token selection across Transformer layers to enhance efficiency and generalization.

6. Acknowledge

The work was supported by the National Natural Science Foundation of China under Grant 62301310.

References

[1] Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your vit but faster, 2023. 3

[2] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 1

[3] 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 (NeurIPS)*, 2022. 2, 5, 6

[4] E Dataset. Novel datasets for fine-grained image categorization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPR Workshops)*, 2011. 5

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2018. 1

[6] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020. 5

[7] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021. 3

[8] Yuntao Du, Siqi Luo, Yi Xin, Mingcai Chen, Shuai Feng, Mujie Zhang, and Chonngjun Wang. Multi-source fully test-time adaptation. *Neural Networks*, 181:106661, 2025. 1

[9] Mohsen Fayyaz, Soroush Abbasi Koohpayegani, Farnoush Rezaei Jafari, Sunando Sengupta, Hamid Reza Vaezi Joze, Eric Sommerlade, Hamed Pirsiavash, and Jürgen Gall. Adaptive token sampling for efficient vision transformers. In *European Conference on Computer Vision*, pages 396–414. Springer, 2022. 3

[10] Timnit Gebru, Jonathan Krause, Yilun Wang, Duyun Chen, Jia Deng, and Li Fei-Fei. Fine-grained car detection for visual census estimation. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2017. 5

[11] 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 *Proceedings of the International Conference on Machine Learning (ICML)*, 2019. 1, 2, 5, 6

[12] 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. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2022. 2, 5, 6

[13] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022. 1

[14] Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge Belongie, Bharath Hariharan, and Ser-Nam Lim. Visual prompt tuning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2022. 1, 2, 5, 6

[15] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5

[16] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13): 3521–3526, 2017. 2, 3

[17] Dongze Lian, Daquan Zhou, Jiashi Feng, and Xinchao Wang. Scaling & shifting your features: A new baseline for efficient model tuning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. 2, 5, 6

[18] Ting Liu, Xuyang Liu, Siteng Huang, Liangtao Shi, Zunnan Xu, Yi Xin, Quanjun Yin, and Xiaohong Liu. Sparse-tuning: Adapting vision transformers with efficient fine-tuning and inference. *arXiv preprint arXiv:2405.14700*, 2024. 1

[19] Xuyang Liu, Ting Liu, Siteng Huang, Yi Xin, Yue Hu, Long Qin, Donglin Wang, Yuanyuan Wu, and Honggang Chen. M2ist: Multi-modal interactive side-tuning for efficient referring expression comprehension. *IEEE Transactions on Circuits and Systems for Video Technology (TCSVT)*, 2025. 1

[20] Siqi Luo, Yi Xin, Yuntao Du, Zhongwei Wan, Tao Tan, Guangtao Zhai, and Xiaohong Liu. Enhancing test time adaptation with few-shot guidance. *arXiv preprint arXiv:2409.01341*, 2024. 1

[21] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *2008 Sixth Indian conference on computer vision, graphics & image processing*, 2008. 5

[22] OpenAI. Gpt-4 technical report. *arXiv preprint arXiv:2304.10592*, 2023. 1

[23] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019. 5

[24] Qi Qin, Le Zhuo, Yi Xin, Ruoyi Du, Zhen Li, Bin Fu, Yiting Lu, Jiakang Yuan, Xinyue Li, Dongyang Liu, et al. Lumina-image 2.0: A unified and efficient image generative framework. *arXiv preprint arXiv:2503.21758*, 2025. 1

[25] Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. Dynamicvit: Efficient vision transformers with dynamic token sparsification. *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. 3

[26] Tal Ridnik, Emanuel Ben-Baruch, Asaf Noy, and Lihi Zelnik-Manor. Imagenet-21k pretraining for the masses. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*, 2021. 5

[27] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023. 1

[28] Grant Van Horn, Steve Branson, Ryan Farrell, Scott Haber, Jessie Barry, Panos Ipeirotis, Pietro Perona, and Serge Belongie. Building a bird recognition app and large scale dataset with citizen scientists: The fine print in fine-grained dataset collection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 5

[29] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. *California Institute of Technology*, 2011. 5

[30] Hongjie Wang, Bhishma Dedhia, and Niraj K Jha. Zero-tprune: Zero-shot token pruning through leveraging of the attention graph in pre-trained transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16070–16079, 2024. 3

[31] Yi Xin, Junlong Du, Qiang Wang, Zhiwen Lin, and Ke Yan. Vmt-adapter: Parameter-efficient transfer learning for multi-task dense scene understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2024. 1

[32] Yi Xin, Junlong Du, Qiang Wang, Ke Yan, and Shouhong Ding. Mmap : Multi-modal alignment prompt for cross-domain multi-task learning. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2024. 1

[33] Yi Xin, Siqi Luo, Xuyang Liu, Haodi Zhou, Xinyu Cheng, Christina E Lee, Junlong Du, Haozhe Wang, MingCai Chen, Ting Liu, et al. V-petl bench: A unified visual parameter-efficient transfer learning benchmark. *Advances in neural information processing systems*, 37:80522–80535, 2024. 1

[34] 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. 2, 5

[35] Yi Xin, Juncheng Yan, Qi Qin, Zhen Li, Dongyang Liu, Shicheng Li, Victor Shea-Jay Huang, Yupeng Zhou, Renrui Zhang, Le Zhuo, et al. Lumina-mgpt 2.0: Stand-alone autoregressive image modeling. *arXiv preprint arXiv:2507.17801*, 2025. 1

[36] Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2022. 1, 2, 5, 6

[37] Xiaohua Zhai, Joan Puigcerver, Alexander Kolesnikov, Pierre Ruyssen, Carlos Riquelme, Mario Lucic, Josip Djolonga, Andre Susano Pinto, Maxim Neumann, Alexey Dosovitskiy, et al. A large-scale study of representation learning with the visual task adaptation benchmark. *arXiv preprint arXiv:1910.04867*, 2019. 5

[38] Yuanhan Zhang, Kaiyang Zhou, and Ziwei Liu. Neural prompt search. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2024. 2

[39] Zhi Zhang, Qizhe Zhang, Zijun Gao, Renrui Zhang, Ekaterina Shutova, Shiji Zhou, and Shanghang Zhang. Gradient-based parameter selection for efficient fine-tuning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 1, 2, 3, 5, 6