Gradient-based Parameter Selection for Efficient Fine-Tuning

Supplementary Material

A. Details of experiments

A.1. Baseline description

Vision Transformer (ViT) As a transformer-based visual model, ViT [14] has been widely adopted in various visual tasks. Most of the experiments are conducted on pre-trained ViT architecture in this paper. Given an input image $I \in \mathbb{R}^{H\times W\times 3}$, before feeding the image into the Transformer, the image is partitioned into M patches and appended a [CLS] token for classification purposes, resulting in final input $x \in \mathbb{R}^{(M+1)\times d}$ where d is the dimension of the features. The Transformer typically consists of multiple blocks and each block contains a Multi-head Attention layer (MHA) and two MLP layers[82].

Adapter Work in [36] proposed the Adapter method, which inserts multiple trainable layers (termed as Adapter) into the pre-trained Transformer encoder. Only the Adapter is updated during the fine-tuning stage. These layers can be inserted after either the Multi-head Attention layer or the MLP layer. Adapter comprises two projection matrices, one W^{down} for dimension reduction and the other W^{up} for feature reconstruction to the original dimension. Specifically, given the input $x \in \mathbb{R}^{(M+1)\times d}$, the output of the Adapter is

$$y = \left[W^{\text{up}}\, \phi\left(W^{\text{down}}\, x^T\right)\right]^T \tag{7}$$

where $W^{\text{up}} \in \mathbb{R}^{d'\times d}$, $W^{\text{down}} \in \mathbb{R}^{d\times d'}$ (where $d' \ll d$), and ϕ is a nonlinear activation function.

Prompt Visual prompt tuning (VPT) introduces learnable parameters ($i.e.$, prompts) into the input space [43]. When fine-tuning downstream tasks, the backbone is fixed, and just tuning these prompts. Formally, the given input $x \in \mathbb{R}^{(M+1)\times d}$ is concatenated with m introduced prompts $p \in \mathbb{R}^{m\times d}$. The final combined input is

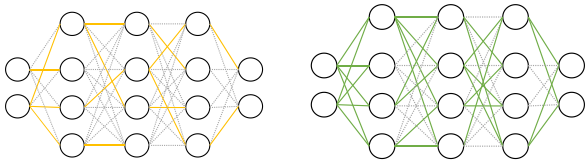
$$x' = [x; p] \tag{8}$$

where $x' \in \mathbb{R}^{(M+1+m)\times d}$ will be feed into the Transformer. There are two versions of VPT, namely VPT-shallow and VPT-deep. The former introduces learnable prompts solely into the input space of the first layer, whereas the latter integrates them into each layer's input space.

Scale and shift feature SSF attempts to scale and shift the features between the layers of the pre-trained model by adding a linear transform layer [58]. During fine-tuning the downstream tasks, only the linear transform layers are updated while the backbone remains frozen. The transform layer consists of two components, scale factor $\gamma \in \mathbb{R}^d$ and shift factor $\beta \in \mathbb{R}^d$, for feature transformation. To be specific, given the input $x \in \mathbb{R}^{(M+1)\times d}$, the output is calculated by

$$y = \gamma \odot x + \beta \tag{9}$$

where $y \in \mathbb{R}^{(M+1)\times d}$, $\odot$ is the dot product.



(a) One input connection (b) Two input connections

Figure 7. Different number of connections with highest gradient value among all input connections per neuron, such as (a) selecting only one input connection per neuron. (b) two input connections are selected per neuron.

A.2. The number of parameters on different tasks

For each neuron in the network, our GPS method selected at least one of the connections (weight or parameter) with the highest gradient value, among the input connections of the neuron, as shown in Fig. 7(a). For downstream tasks that need more learnable parameters to better fit the data, such as those tasks with dissimilar data distributions from the upstream dataset (such as NABirds) or larger amounts of data (such as CIFAR-100), our method can be easily extended by introducing more learnable parameters. Specifically, for each neuron, we can select multiple input connections with the highest gradient values instead of limiting them to just one, as shown in Fig. 7(b). Tab. 7 show the detailed statics on the number of parameters that are selected in our paper. For most of the tasks in this paper, we just select one of the connections. We also explore the relationship between the number of connections and the number of learnable parameters. As shown in Fig. 8, with the increase in the number of selected connections with the highest gradient value among the input connections per neuron, the number of learnable parameters linear ascent.

A.3. Parameters distribution of *Net* selection

In contrast to our approach, a simple approach is to select the parameters for a specific task by selecting a certain percentage of parameters with the highest gradient from the en-

Dataset	Params. (M)	Conne.s	Dataset	Params. (M)	Conne.s	Dataset	Params. (M)	Conne.s
CUB-200-2011	0.47	2	Pets	0.23	1	DMLab	0.20	1
NABirds	1.35	10	SVHN	0.20	1	KITTI/distance	0.20	1
Oxford Flowers	0.29	1	Sun397	0.55	1	dSprites/loc	0.21	1
Stanford Dogs	0.30	1	Patch Camelyon	0.30	2	dSprites/ori	0.21	1
Stanford Cars	1.07	10	EuroSAT	0.20	1	SmallNORB/azi	0.21	1
CIFAR-100*	0.29	1	Resisc45	0.24	1	SmallNORB/ele	0.20	1
Caltech101	0.29	1	Retinopathy	0.20	1	CIFAR-100	0.58	5
DTD	0.24	1	Clevr/count	0.30	1	CIFAR-100 (Swin)	0.82	5
Flowers102	0.29	1	Clevr/distance	0.20	1	CIFAR-100 (ConvNeXt)	0.78	5

Table 7. The number of learnable parameters and connections across all tasks. CIFAR-100* is a subset of CIFAR-100 in VTAB benchmark. In bracket is the model architecture, without bracket represents the one fine-tuned on ViT-B/16. Params. means the learnable parameters and the Conne. represents the number of selected input connections for each neuron in the network.

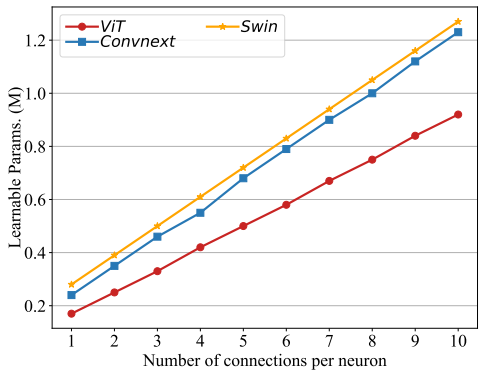


Figure 8. The number of learnable parameters with the different number of connections on VIT-B/16, Swin and Convnext archite. The learnable parameters do not contain the task-specific head.

Method \ Dataset	ImageNet -1K (↑)	ImageNet -A (↑)	ImageNet -R (↑)	ImageNet -C (↓)
Full [43]	83.58	34.49	51.29	46.47
Linear [43]	82.04	33.91	52.87	46.91
Bias [92]	82.74	42.12	55.94	41.90
Adapter [36]	82.72	42.21	54.13	42.65
VPT-Shallow [43]	82.08	30.93	53.72	46.88
VPT-Deep [43]	82.45	39.10	53.54	43.10
SSF [58]	83.10	45.88	56.77	**41.47**
GPS	**83.91**	**46.11**	**57.00**	42.04

Table 8. Performance comparisons on the ImageNet with different model architectures.

tire network [30]. However, as shown in Figs. 9(a) to 9(e), most of the selected parameters are located in the upper layers, specifically block 12 and block 11. As a result, the network is primarily focused on fine-tuning abstract features while lacking the ability to fine-tune detailed information from shallower layers. Our approach addresses this challenge by carefully selecting the input connections for each individual neuron, resulting in our selected parameters being evenly distributed on the whole network, as shown in Fig. 9(f).

B. Additional experiments

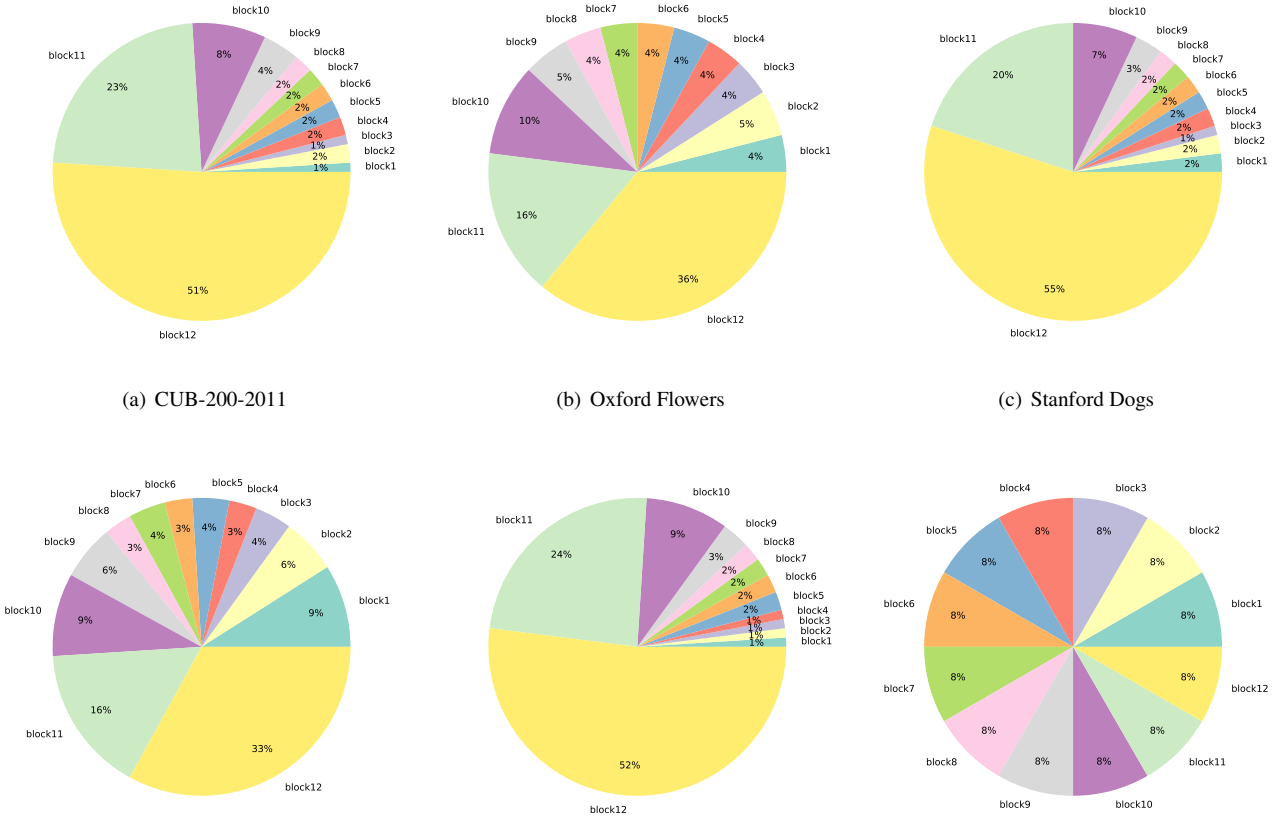
B.1. Robustness and OOD datasets

In addition to standard classification tasks, we further analyze the robustness and OOD generalization ability of GPS. Based on the Imagenet-A, ImageNet-R, and ImageNet-C datasets, we first fine-tune the model on ImageNet-1K, and then test the fine-tuned model on the three datasets respectively. The results are shown in Tab. 8. GPS not only achieves the best performance on the standard ImageNet-1K classification task but also achieves good performance in robustness and generalization tests. Among them, GPS achieves the best results on ImageNet-A and ImageNet-R, outperforms the previous optimal SSF by 0.23%, reflecting the strong stability and generalization ability of our method. On ImageNet-C, GPS performs slightly worse, lagging behind SSF, but still higher than addition-based Adapter and VPT. This result indicates that our method can quickly adapt to the data distribution of downstream tasks, but it needs to be improved in anti-interference.

B.2. More experiments on different architecture

As mentioned in the main body of our paper, our method is model-agnostic, we further compare GPS with other fine-tuning methods across ViT-B/16, Swin-B, and ConvNeXt-B architectures on the ImageNet-1k [10] and CIFAR-100 [51] datasets.

CIFAR-100 As shown in Tab. 10, unlike FGVC and VTAB, GPS and other efficient tuning methods have difficulty in achieving competitive performance as full tuning on CIFAR-100. This may be due to that CIFAR-100 contains more training data, allowing all parameters of the entire model to be adequately trained, which seriously reduces the advantages of efficient fine-tuning methods. However,



(a) CUB-200-2011 (b) Oxford Flowers (c) Stanford Dogs

(d) Stanford Cars (e) NABirds (f) GPS

Figure 9. Distribution of the parameter over the whole network ViT-B/16 on 6 FGVC dataset (CUB-200-2011, Oxford Flowers, Stanford Dogs, Stanford Cars and NABirds). Selecting the 1% parameters with the highest gradient value from the whole network, instead of our method selecting at least one of the connections among all input connections per neuron. In contrast to this method, our GPS has the same distribution over different downstream tasks (f).

Method \ Dataset	CUB-200 -2011	NABrids	Oxford Flowers	Stanford Dogs	Stanford Cars	Mean Acc.	Mean Params. (M)	Mean Params. (%)
ViT-B/16 + Full	87.3	82.7	98.8	89.4	84.5	88.54	85.98	100.00
ViT-B/16 + Linear	85.3	75.9	97.9	86.2	51.3	79.32	0.18	0.21
ViT-B/16 + SSF	89.5	85.7	99.6	89.6	89.2	90.72	0.39	0.45
ViT-B/16 + GPS (Ours)	**89.9**	**86.7**	**99.7**	**92.2**	**90.4**	**91.78**	0.66	0.77
Swin-B + Full	90.7	**89.8**	99.5	88.9	**93.2**	92.42	86.98	100.00
Swin-B + Linear	90.6	86.8	99.2	88.3	74.6	87.90	0.24	0.28
Swin-B + SSF	90.5	88.4	**99.7**	88.7	90.4	91.54	0.49	0.56
Swin-B + GPS (Ours)	**90.8**	88.9	**99.7**	**92.7**	90.7	**92.56**	0.83	0.95
ConvNeXt-B + Full	**91.2**	**90.4**	99.6	89.9	**94.1**	93.04	87.81	100.00
ConvNeXt-B + Linear	90.6	86.9	99.3	89.7	73.5	88.00	0.24	0.28
ConvNeXt-B + SSF	90.8	89.0	**99.7**	90.4	92.5	92.48	0.50	0.56
ConvNeXt-B + GPS (Ours)	91.0	89.6	**99.7**	**93.7**	92.6	**93.32**	0.79	0.90

Table 9. Performance comparisons on FGVC benchmark with different model architectures.

Architecture	ViT-B/16		Swin-B		ConvNeXt-B	
	Acc.	Params.(%)	Acc.	Params.(%)	Acc.	Params.(%)
Full [43]	93.82	100.00	**93.85**	100.00	**94.14**	100.00
Linear [43]	88.70	0.09	89.27	0.12	89.20	0.12
Bias [92]	93.39	0.21	92.19	0.28	92.80	0.27
Adapter [36]	93.34	0.36	92.49	0.38	92.86	0.52
VPT-Shallow [43]	90.38	1.07	90.02	0.15	-	-
VPT-Deep [43]	93.17	1.43	92.62	0.81	-	-
SSF [58]	93.99	0.33	93.06	0.43	93.45	0.42
GPS (Ours)	**94.02**	0.68	93.55	0.96	93.58	0.90

Table 10. Performance comparisons on the CIFAR-100 with different model architectures.

Architecture	ViT-B/16		Swin-B		ConvNeXt-B	
	Acc.	Params.(%)	Acc.	Params.(%)	Acc.	Params.(%)
Full [43]	83.58	100.00	**85.20**	100.00	**85.80**	100.00
Linear [43]	82.04	0.89	83.25	1.17	84.05	1.16
Bias [92]	82.74	1.00	83.92	1.32	84.63	1.31
Adapter [36]	82.72	1.16	83.82	1.43	84.49	1.54
VPT-Shallow [43]	82.08	1.06	83.29	1.19	-	-
VPT-Deep [43]	82.45	1.42	83.44	1.85	-	-
SSF [58]	83.10	1.12	84.40	1.47	84.85	1.44
GPS (ours)	**83.91**	1.37	84.43	1.96	84.87	1.90

Table 11. Performance comparisons on the ImageNet-1k with different model architectures.

GPS still outperforms all previous parameter-efficient tuning methods (Bias, Adapter, VPT, and SSF) and reduces the gap with full fine-tuning to less than 0.5% on all architectures, which further demonstrates the adaptability of our approach to different models.

ImageNet-1k Similar to the results on CIFAR-100, ImageNet-1K contains more training data, which makes it harder for parameter-efficient fine-tuning algorithms to achieve the same accuracy as full fine-tuning, as shown in Tab. 11. However, GPS still outperforms full fine-tuning by 0.33% on ViT structure, and outperforms the previous SOTA method SSF on Swin and ConvNeXt structures respectively, which further shows the generalization of GPS to different model structures.

FGVC As mentioned in the main body of our paper, our method achieves the best result on FGVC benchmark. Tab. 9 shows the full results of Tab. 4 in the main body. Among all three model architectures, GPS consistently outperforms all other baselines, demonstrating its model-agnostic advantage.

B.3. Data-efficient tuning

Recent advances in large foundation model fine-tuning have shown considerable promise in reaching state-of-the-art performance on various tasks. However, in order to reach high accuracy, these methods often need significant volumes of training data, which may be time-consuming and costly to obtain. Here we demonstrate that our method is data efficient, that is, with such a few-shot setting, our method requires only a small amount of training data for tuning to achieve outstanding results that other approaches do not. Specifically, we fine-tune the ViT-B/16 by selecting only k samples for each class in the ImageNet dataset to form a few-shot training set. The value of k and the accuracy of predicted results are illustrated in Fig. 10, which demonstrates the excellent data efficiency of our method especially in extreme cases like k=1.

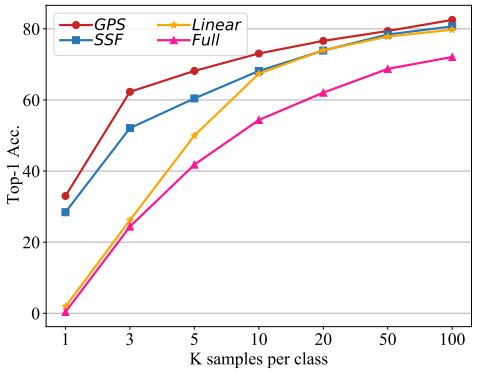


Figure 10. The performance comparison of different fine-tuning methods under k values for each class on ImageNet dataset. Our method is always above the curve of the others. The advantage of our approach is particularly evident in extreme cases where data is extremely scarce (e.g., k=1).

B.4. Random seed for impacts of different selection schemes and ablations

We conduct experiments with three random seeds to investigate the robustness of our method. As shown in Tab. 12, Our parameter selection method significantly outperforms the other methods with small randomness. Tab. 12 is a supplementary of Tab. 6 in the main body of our paper.

C. Visualizations

C.1. Semantic segmentation

As mentioned in the main body of our paper, our approach demonstrates highly promising results in the field of semantic segmentation. We apply our method on the pre-trained strong segmentation model (SAM) [50] and fine-tune on a medical segmentation task – polyp segmentation [41]. Here, we present more case visualizations, which could directly show the effectiveness of our method, as shown in Fig. 11.

		CUB	NAbirds	Flowers	Cars	Dogs
(a)	Net	86.86 ± 0.21	86.55 ± 0.03	99.62 ± 0.01	89.65 ± 0.12	91.32 ± 0.07
	Layer	87.30 ± 0.13	86.79 ± 0.08	99.64 ± 0.01	90.03 ± 0.13	91.90 ± 0.11
(b)	Net Random	86.60 ± 0.10	85.98 ± 0.07	99.61 ± 0.01	89.10 ± 0.12	91.34 ± 0.12
	Neuron Random	87.17 ± 0.15	86.02 ± 0.10	99.62 ± 0.01	89.52 ± 0.23	91.82 ± 0.23
	Magnitude	87.29 ± 0.12	85.99 ± 0.08	99.62 ± 0.00	89.29 ± 0.02	91.30 ± 0.02
(c)	Head+CE	87.05 ± 0.19	86.20 ± 0.14	99.64 ± 0.01	89.25 ± 0.09	91.29 ± 0.01
	GPS	**88.07** ± 0.11	**86.64** ± 0.03	**99.69** ± 0.01	**90.10** ± 0.10	**92.30** ± 0.10

Table 12. Impacts of different selection schemes and ablations. (a) Different selection levels. (b) Different selection criteria. (c) Different ways to calculate gradients.

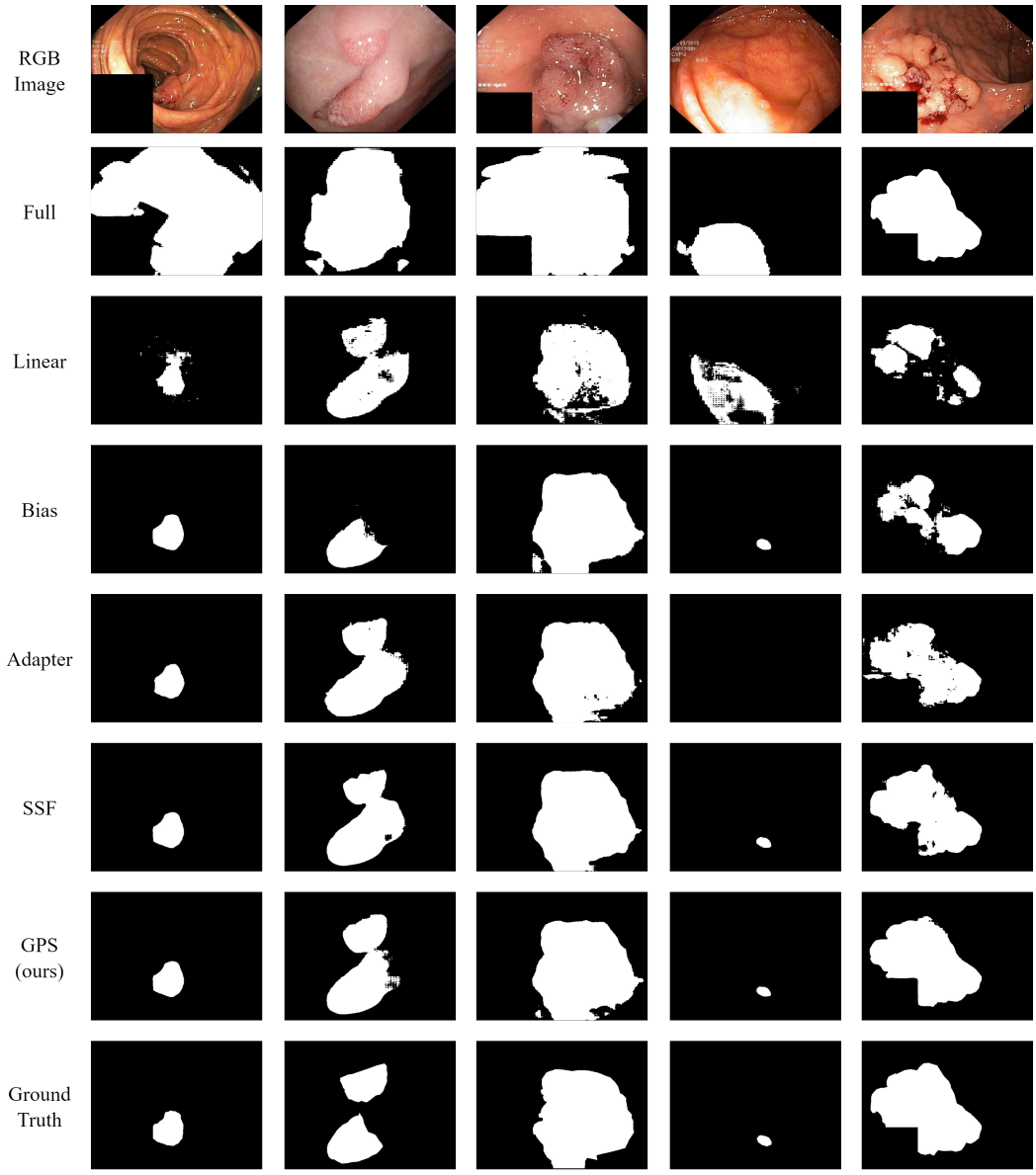
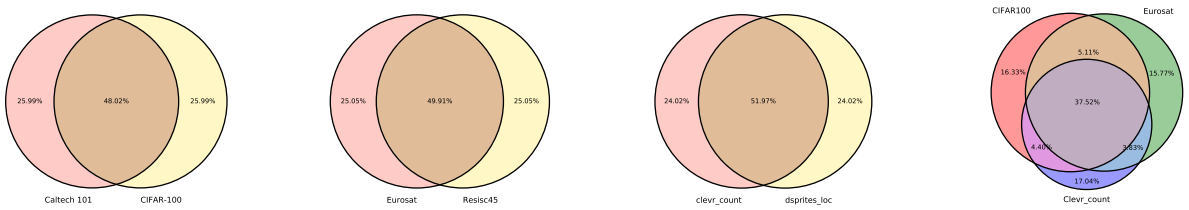
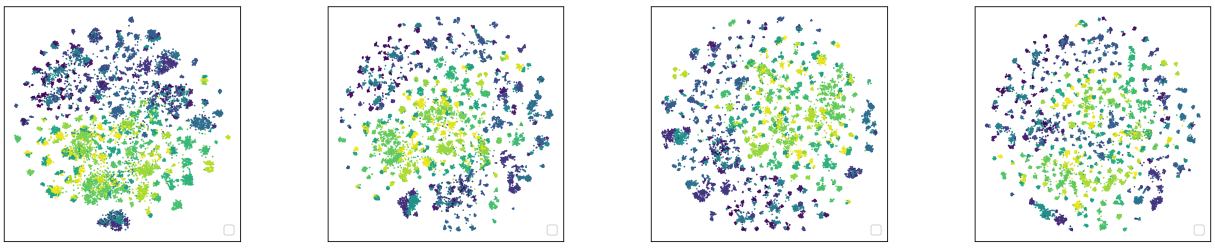


Figure 11. **The Visualization Result of Polyp Segmentation.** Here GT means ground truth. As illustrated in this figure, although SAM and other methods can identify some polyp structures in the image, the result is not accurate. By using GPS, our approach elevates the performance with SAM.



(a) (b) (c) (d)

Figure 12. The overlap of the selected parameters across different tasks. Overlap is determined based on parameter position. If selected parameters share the same position in the network, they are considered to have overlap. We test on the ViT-B/16 with following tasks: (a) Cifar100 and Caltech101; (b) Eurosat and Resisc45; (c) Clevr/count and Dsprites/loc; (d) Cifar100, Eurosat and Clevr/count.



(a) Linear Probing (b) SSF (c) GPS (ours) (d) Full fine-tuning

Figure 13. t-SNE visualization of different fine-tuning methods, including linear probing, SSF, GPS, full fine-tuning.

C.2. Distribution of selected parameters across various tasks

As we select different subsets of parameters from the original model for different downstream tasks, a normal question is how different the distribution of the selected parameters is across different tasks. we test on Vit-B/16 with 6 downstream tasks from VTAB (two from Natural, two from Specialized and the other two from Structured). As shown in Fig. 12, the chosen parameters exhibit a tendency towards 2/3 shared parameters and 1/3 task-specific parameters, despite the dissimilar data distribution of downstream tasks. This is due to our selection scheme, which makes the parameters evenly distribute on the whole network and thus the parameters from shallow layers tend to share parameters as similar findings from the field of multi-task learning [46, 69, 80].

C.3. Feature distribution

On the NABirds datasets, we use t-SNE to visualize the feature distribution of different fine-tuning methods. The results of all comparison methods are obtained based on the ViT-B/16 pre-trained on ImageNet-21k. The visualization results are illustrated in Fig. 13. Feature clustering results using our GPS are superior to those with linear probing, SSF, and full fine-tuning.

D. Details of the evaluation datasets

The statistic of all datasets used in this paper is shown in Tab. 13.

D.1. Image classification

FGVC Fine-Grained Visual Classification (FGVC) benchmark includes 5 downstream tasks, which are CUB-200-2011 [84], NABirds [81], Oxford Flowers [67], Stanford Dogs [47] and Stanford Cars [23]. Each one contains more than 100 classes and a few thousand images. We directly use the public splits if one contains, otherwise, we follow the splits in [43].

VTAB-1k Visual Task Adaptation Benchmark [93] contains 19 visual classification tasks which are grouped into 3 sets: (1) Natural – tasks with natural images captured by standard cameras; (2) Specialized – tasks with images captured via specialized equipment, e.g., medical camera or satellite sensor; (3) Structured – tasks with images synthesized from simulated environments, which require geometric comprehension like object counting and depth estimation. Each one contains only 1000 training examples while

Dataset	Description	#Classes	Train	Val	Test
Fine-Grained Visual Classification (FGVC)					
CUB-200-2011 [84]	Bird recognition	200	5,394	600	5,794
NABirds [81]	Bird recognition	555	21,536	2,393	24,633
Oxford Flowers [67]	Flower recognition	102	1,020	1,020	6,149
Stanford Dogs [47]	Dog recognition	120	10,800	1,200	8,580
Stanford Cars [23]	Car classification	196	7,329	815	8,041
Visual Task Adaptation Benchmark (VTAB-1k) [93]					
CIFAR-100 [51]	Natural	100	800/1000	200	10,000
Caltech101 [16]		102			6,084
DTD [8]		47			1,880
Flowers102 [67]		102			6,149
Pets [68]		37			3,669
SVHN [66]		10			26,032
Sun397 [89]		397			21,750
Patch Camelyon [83]	Specialized	2	800/1000	200	32,768
EuroSAT [32]		10			5,400
Resisc45 [7]		45			6,300
Retinopathy [25]		5			42,670
Clevr/count [44]	Structured	8	800/1000	200	15,000
Clevr/distance [44]		6			15,000
DMLab [2]		6			22,735
KITTI/distance [24]		4			711
dSprites/location [64]		16			73,728
dSprites/orientation [64]		16			73,728
SmallNORB/azimuth [54]		18			12,150
SmallNORB/elevation [54]		9			12,150
General Image Classification Datasets					
CIFAR-100 [51]	General images	100	50,000	-	10,000
ImageNet-1K [10]		1,000	1,281,167	50,000	150,000
Robustness and Out-of-Distribution Datasets					
ImageNet-A [35]	Robustness & OOD	200	-	-	7,500
ImageNet-R [34]		200	-	-	30,000
ImageNet-C [33]		1,000	-	-	75×50,000
Cross-domain Semantic Segmentation Dataset					
Kvasir-SEG [41]	Polyp Segmentation	2	880	-	120

Table 13. Detailed statistics of the datasets evaluated on our work. We follow the VPT [43] for train/val split. This table is partially borrowed from VPT[43] and SSF [58].

a large number of test images (i.e. over 20,000 on average).

CIFAR-100 CIFAR-100 [51] is a widely used general image classification task. It contains 50,000 training and 10,000 test images with 100 categories.

ImageNet-1K ImageNet-1K [10] is the most commonly utilized subset of ImageNet for object classification, encompassing 1000 classes and featuring a training set of 1,281,167 images, a validation set of 50,000 images, and a test set of 100,000 images.

D.2. Semantic segmentation

Polyp segmentation We select kvasir-SEG [42] for polyp segmentation task. We follow the settings in Medico automatic polyp segmentation· task at mediaeval 2020 [41] with a train-valid ratio of 880:120.

D.3. Robustness and OOD

ImageNet-A ImageNet-A [35] contains 200 classes, which is selected from ImageNet-1K (1000 classes). All samples are real-world adversarial samples that caused the ResNet model to produce erroneous classifications.

ImageNet-R ImageNet-R [34] contains art, graffiti, sculptures, tattoos, toys, cartoons, paintings, embroidery, deviantart, graphics, patterns, plastic objects, origami, plush objects, sketches, and video game renditions from ImageNet classes.

ImageNet-C ImageNet-C [33] is an open-source collection of algorithmically generated corruptions, such as blur and noise, that have been applied to the ImageNet test set.

E. Extended related work

E.1. Visual parameter efficient fine-tuning

In the field of computer vision, current work endeavors to pre-train larger models [11, 14, 61, 73, 99, 100] on extensive datasets, followed by fine-tuning diverse downstream tasks to achieve superior performance and faster convergence. Conventional arts set all the network parameters learnable and adapt them to the target tasks. However, as foundation models become increasingly large and the number of downstream tasks increases, it becomes impractical due to the significant computational and storage requirements that it entails. Parameter-efficient fine-tuning (PEFT) methods are proposed to alleviate such a burden, which tunes only a tiny portion of the parameters. The general PEFT can be categorized into addition-based and selection-based methods.

Addition-based methods introduce additional parameters to the pre-trained backbone. Adapter methods keep most of the parameters in the model frozen and update only small-scale injected parameters. Bottleneck-structured adapters [1, 36, 69, 70, 75, 76, 78, 80, 86, 94] adopt a residual pathway to leverage both original and task-specific knowledge by learning down-projection and up-projection with a nonlinear activation. Others [63] propose a hypernetwork to generate model weights or decompose the dense weighted matrix into the low-rank matrix to reduce parameters [46]. Instead of introducing extra modules, prompt methods [12, 22, 39, 45, 57, 59, 60] wrap the input with context. A representative work VPT [43] prepend learnable prompts to the input tokens before feeding it into each Transformer block. VPT includes two variants VPT-Shallow and VPT-Deep associated with the number of inserted layers. VPT-Shallow simply prepends prompts to the first transformer layer while VPT-Deep prepends prompts to all the layers. However, it's inflexible when applying the method to new tasks since it relies on hand-crafted prompt length selection. Apart from the adapter and prompt tuning, a recent study SSF [58] introduces two learnable vectors to scale and shift the feature map in each transformer operation and achieves promising results. These extra parameters will lead to a substantial increase in computational overhead and hinder the rate of convergence. Our method solves these issues without adding parameters or changing the network topology so it can effectively alleviate such problems.

Selection-based methods [26, 92, 98] do not introduce any new parameters but directly select part of the parameters to be optimized without modifying the intrinsic architecture of the model. Bitfit [92] only fine-tunes bias vectors in the pre-trained model. Other methods only fine-tune the top-K layers [36] or the last linear layer [43] with other layers freeze. Despite efficiency, they suffer a significant accuracy drop compared to the full fine-tuning since the manually specified parameters tend to be a non-optimal solution. Our gradient-based parameter selection method falls into this category. Since the gradient can serve as a tool for determining parameter significance, our method is intuitive but surprisingly effective.

E.2. Subset network training

Standard pruning technique [19, 28, 29, 52, 56, 88] naturally uncovers subnetworks whose initializations made them capable of training effectively. The lottery ticket hypothesis [17] articulate that subnetworks can reach test accuracy comparable to the original network. Drawing from the theory, fine-tuning methods based on subset network are widely studied. SpotTune [27] designs a policy network to make routing decisions for subset networks. Child-tuning [90] iteratively updates a subset of parameters by masking out the gradients of the non-child network during the backward process. However, the computing overhead led by hyper networks or iterative parameter selection makes none of these methods parameter-efficient. We fix the position of parameters that will be updated by simple gradient weights sorting before training which makes our method parameter efficient.

F. Disscussion

Why sub-network? There is a lot of research in the field of neural network pruning, where researchers aim to identify the importance of the parameters in a network and eliminate some unnecessary parameters without perfor-

mance deterioration (about 90% parameters of the model are pruned) [5, 9, 55, 71]. Motivated by this, we posit the existence of a sub-network containing crucial parameters that can be fine-tuned for optimal performance on downstream tasks.

Magnitude or gradient? In contrast to the approaches of identifying the importance of the parameters in [5, 9, 55, 71], which rely on weight magnitude to determine parameter importance, our method identifies parameter importance based on gradient values. An important difference between gradient and magnitude is that the gradient-based method is task-specific, as the gradient is calculated by the backpropagation of the loss for a specific task, while the magnitude-based method use a set of same parameters for all downstream tasks. However, our ablation study in the main body has shown gradient-based method perform bwtter and the Fig. 12 also show that each task has it own task-specific parameters.

G. Limitations and societal impacts

Limitations Several studies [46, 69, 80] have demonstrated that certain similar tasks can be optimized together through parameter sharing, resulting in improved performance across all individual tasks. However, our work focuses on selecting distinct parameters for various tasks. Although we already tune affordable parameters, we do not fully exploit the potential of parameter sharing across different tasks. Therefore, we posit that our work can be extended to a multitask setting, where tasks share tuning parameters and thus further reduce the total number of learnable parameters.

Societal impacts Our method can effectively fine-tune pre-trained models for downstream tasks by adjusting less than 1% of the network's parameters. This is particularly beneficial when dealing with large pre-trained models and multiple downstream tasks, as it saves computational resources, memory costs, and reduces carbon emissions. Our approach maintains the model's original structure without introducing any additional parameters during both the training and inference stages, distinguishing it from other methods. However, similar to other fine-tuning approaches, our method relies on a pre-trained model. If this upstream pre-trained model is trained on illicit data, it may also violate the use of fine-tuning methods.