

# Parameter-efficient is not Sufficient: Exploring Parameter, Memory, and Time Efficient Adapter Tuning for Dense Predictions

Dongshuo Yin\*  
University of Chinese Academy of  
Sciences  
Beijing, China  
yindongshuo19@mailsucas.ac.cn

Xueting Han†  
Microsoft Research Asia  
Beijing, China  
chrihan@microsoft.com

Bin Li  
Alibaba Group  
Beijing, China  
zhuyi.lb@alibaba-inc.com

Hao Feng  
Alibaba Group  
Beijing, China  
yuanning.fh@alibaba-inc.com

Jing Bai  
Microsoft Research Asia  
Beijing, China  
jbai@microsoft.com

## Abstract

Pre-training & fine-tuning is a prevalent paradigm in computer vision (CV). Recently, parameter-efficient transfer learning (PETL) methods have shown promising performance in adapting to downstream tasks with only a few trainable parameters. Despite their success, the existing PETL methods in CV can be computationally expensive and require large amounts of memory and time cost during training, which limits low-resource users from conducting research and applications on large models. In this work, we propose Parameter, Memory, and Time Efficient Visual Adapter (E<sup>3</sup>VA) tuning to address this issue. We provide a gradient backpropagation highway for low-rank adapters which eliminates the need for expensive backpropagation through the frozen pre-trained model, resulting in substantial savings of training memory and training time. Furthermore, we optimise the E<sup>3</sup>VA structure for CV tasks to promote model performance. Extensive experiments on COCO, ADE20K, and Pascal VOC benchmarks show that E<sup>3</sup>VA can save up to 62.2% training memory and 26.2% training time on average, while achieving comparable performance to full fine-tuning and better performance than most PETL methods. Note that we can even train the Swin-Large-based Cascade Mask RCNN on GTX 1080Ti GPUs with less than 1.5% trainable parameters.

## CCS Concepts

• **Computing methodologies** → **Object detection; Image segmentation.**

\*The work was initiated when the author was interning at Alibaba Group and completed when the author was interning at Microsoft Research Asia.

† Corresponding author.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).

MM '24, October 28–November 1, 2024, Melbourne, VIC, Australia

© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN 979-8-4007-0686-8/24/10

<https://doi.org/10.1145/3664647.3680940>

## Keywords

Dense predictions, Memory-efficient fine-tuning, Visual adapter

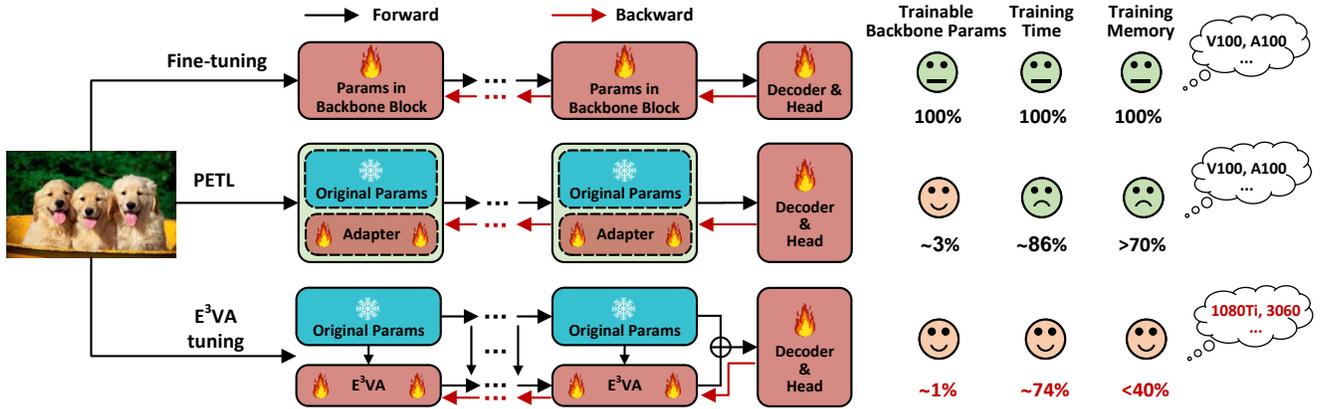
### ACM Reference Format:

Dongshuo Yin, Xueting Han, Bin Li, Hao Feng, and Jing Bai. 2024. Parameter-efficient is not Sufficient: Exploring Parameter, Memory, and Time Efficient Adapter Tuning for Dense Predictions. In *Proceedings of the 32nd ACM International Conference on Multimedia (MM '24)*, October 28–November 1, 2024, Melbourne, VIC, Australia. ACM, New York, NY, USA, 9 pages. <https://doi.org/10.1145/3664647.3680940>

## 1 Introduction

With the development of computer vision (CV), the explosion in model size and capacity has become an unstoppable trend [12, 30, 31, 55, 56, 59, 66]. Most latest large models are transformer-based (e.g., Swin [35], ViT [8], BEiT [54], etc.) and can reach the scale of billions of parameters [63]. Large models have shown tremendous propulsive power in deep learning-based dense prediction vision tasks, including instance segmentation [12], object detection [55], semantic segmentation [54]. However, large models not only bring impressive performance but also massive training and storage costs [3, 35, 43, 47]. It is difficult for users with limited budgets to train or even fine-tune high-quality large models. Meanwhile, cloud service providers (e.g. Google and Amazon) have started to consider the storage costs for massive downstream tasks [18, 49]. In order to reduce the cost of training large models, well-resourced users pre-train state-of-the-art (SOTA) vision models with advanced GPUs and large data resources. After that, users with fewer resources can fine-tune the pre-trained models to achieve impressive performance. However, this traditional "pre-training & fine-tuning" paradigm has its limitations [2, 22, 46, 50, 52]. Fine-tuning still has high hardware requirements as the model size and training memory is not reduced, and new tasks produce the same-sized models as the pre-trained model, which is inefficient for numerous downstream tasks.

Inspired by the success of recent study in Natural Language Processing (NLP) [14, 37, 41, 42, 45, 64], many novel parameter-efficient transfer learning (PETL) methods have recently emerged in CV, including prompt-based [19, 24, 25, 29, 36, 39, 48, 53] and adapter-based [4, 5, 13, 20, 34, 61]. Most of these works have focused on classification tasks, while we comprehensively compare



**Figure 1: Comparison of backpropagation processes, parameters, time and memory of three training paradigms. Left: The backpropagation of full fine-tuning and the existing PETL methods (e.g., adapter-tuning) goes through all parameters, which includes the frozen (❄️) backbone. E<sup>3</sup>VA’s backpropagation highway contains only a tiny number of trainable (🔥) parameters, which avoid gradient backpropagation through the backbone. Right: E<sup>3</sup>VA can significantly save memory and time, allowing for training SOTA large models (e.g. Swin-L with Cascade Mask RCNN) on cheaper GPUs (e.g. GTX 1080Ti).**

the PETL methods on more challenging dense prediction tasks. PETL methods select a small subset of pre-trained parameters or insert extra structures into the backbone and freeze most of the pre-trained backbone parameters during training, which means only those selective or newly added parameters are trainable. PETL can substantially reduce the number of trainable parameters for downstream tasks (even by more than 95%), while maintaining performance comparable to full tuning. However, existing PETL methods are memory and time inefficient. Gradient computation for these trainable parameters still requires backpropagation through the backbone models, resulting in massive training memory and time consumption during training [33, 51]. Sung *et al.* [51] proposes a Ladder Side-Tuning (LST) architecture for NLP tasks (based on T5 model [43]) to reduce the training memory. However, LST is not directly applicable to CV models such as Swin Transformer [35], and it cannot achieve comparable performance to full fine-tuning and other PETL methods.

To address these issues, we propose a novel Parameter-Efficient, Memory-Efficient and Time-Efficient Visual Adapter (E<sup>3</sup>VA) tuning method that establishes a more efficient visual training paradigm. First, we separate adapters from the backbone network rather than plug them into the backbone. This design creates a clear gradient backpropagation highway exclusively for adapters, all trainable parameters are on this highway, preventing backpropagation through the backbone model. Additionally, we adopt a design of parallel adapters instead of stacked ones, aiming to further shorten the length of the backpropagation highway. This not only minimizes the memory required for activation, but also significantly reduces the computational load during backpropagation, consequently saving substantial GPU memory and training time. Figure 1 illustrates the differences between E<sup>3</sup>VA and previous methods. Furthermore, we introduce a dual low-rank structure in E<sup>3</sup>VA adapter to minimize the adapter’s parameters and training memory. To adapt for

visual models, we add downsampling layers into the highway, inheriting parameters from the backbone to accommodate dimension reductions in visual models. Additionally, we integrate the highway into the Feature Pyramid Network (FPN) alongside the backbone and set FPN norms as trainable to optimize E<sup>3</sup>VA-tuning in dense prediction tasks.

To demonstrate the effectiveness and efficiency of E<sup>3</sup>VA, we conduct extensive experiments on MS COCO [28], PASCAL VOC [11] and ADE20K [65] for mainstream dense prediction tasks, including instance segmentation, object detection and semantic segmentation. Experimental results show that E<sup>3</sup>VA can **save up to 62.2% training memory and 26.2% training time on average** compared to the full fine-tuning, while achieving comparable performance to full fine-tuning and better performance than most PETL methods. It is worth noting that, based on results on COCO benchmark, E<sup>3</sup>VA-tuning can tune Swin-Large+Cascade Mask RCNNs on the Tesla P100/GTX 3090 and achieve even better performance than the Swin-Base+Cascade Mask RCNNs which are fully fine-tuned on the 32 GB Tesla V100 (Swin-Large+Cascade Mask RCNNs can’t be fully fine-tuned on 32 GB Tesla V100), which means that E<sup>3</sup>VA enables GPU-starved users to train large models efficiently. Moreover, E<sup>3</sup>VA can alleviate the over-fitting issue in low-resource [10, 41, 60] situations.

## 2 Related Works

### 2.1 Parameter-Efficient Transfer Learning

PETL has received much attention in the NLP field recently, as it can achieve comparable performance of 100% fine-tuning by tuning even less than 1% of the parameters [1, 23, 41, 45, 64]. PETL methods in NLP freeze most of the parameters in the transformer and train a small number of specified parameters. BitFit [62] only tunes the bias terms, Prompt-tuning [32] adds learnable tokens to the input layer, Adapter-tuning [14] adds some trainable bottleneck structures between layers, LoRA [16] injects trainable rank

decomposition matrices into each layer of the transformer architecture, and compacter [23] introduces low-rank structures into the adapter structure to further reduce the number of parameters. After continuous optimization and extensive validation, PETL can even outperform full fine-tuning in NLP. Advanced results in NLP have also brought new thinking to CV.

As a first attempt, Jia *et al.* [19] proposes VPT and demonstrates that the prompt-based PETL approach outperforms full fine-tuning in image classification tasks. Polyhistor [34] introduces PETL in the multi-task setting. Kronecker Adaptation [13] reduces the number of parameters in adapters by introducing the Kronecker product. AdaptFormer [5] utilizes the adapter structure and surpasses VPT in image classification. Convpass [20] replaces the bottleneck structure in LoRA with a CNN structure and improves the performance of PETL on image classification tasks. LoRand [61] employs multi-branch low-rank adapters for adapting dense prediction tasks. In addition, PETL has also been applied to visual-language [21], action recognition [40], dense scene understanding [58], and remote sensing [17] tasks.

## 2.2 Memory-Efficient Transfer Learning

Although PETL methods can save most trainable parameters, they do not reduce much memory requirement and training time. NLP researchers first studied this issue. Since the inserted structures are located inside the backbone models, to calculate gradients for these parameters, the backpropagation still need to go through the large backbone model, making the PETL methods not memory and time efficient [33, 51]. Gradient checkpointing [7] trades extra time for reduced memory by clearing activations of certain layers and recomputing them during a backward pass. Y-tuning [33] learns extra task-specific label representations and fuses them with the output of the backbone model. This design avoids backpropagation through pre-trained model, but it does not achieve good performance and is challenging to extend to tasks other than classification. LST [51] utilizes lightweight transformer structures pruned from the backbone as the side network. It's designed for NLP tasks and can not be directly used in CV models such as Swin Transformer, and it cannot achieve comparable performance to full fine-tuning and other PETL methods. In contrast, we design the adapter highway to retain the expressiveness of adapter tuning while saving memory and time cost. We propose several novel designs to improve accuracy and training efficiency, which include special designs for CV models. It achieves comparable performance to full fine-tuning and other PETL methods.

## 3 Methods

We introduce the proposed approach in four parts, including training paradigms, overall framework of E<sup>3</sup>VA, and propagation comparisons with existing tuning methods.

### 3.1 Preliminaries

For dataset  $\mathcal{D} = \{(X_i, Y_i)\}_{i=1}^N$ , the loss  $\mathcal{L}$  and optimization formula of full fine-tuning, adapter-tuning, E<sup>3</sup>VA-tuning can be written as follows:

### Full Fine-tuning

$$\mathcal{L}(\mathcal{D}, \phi) = \sum_{i=1}^N \text{loss}(f_{\phi}(x_i), y_i), \quad (1)$$

$$\phi \leftarrow \arg \min_{\phi} \mathcal{L}(\mathcal{D}, \phi), \quad (2)$$

where  $\phi$  is the parameters of the model,  $\text{loss}$  is the loss function and  $f(\cdot)$  is the forward propagation function.

### Adapter-tuning

The parameters in adapter-tuning [14] can be divided into fixed parameters  $\phi_F$  and trainable parameters  $\Omega$ , where trainable parameters  $\Omega$  can be further divided into  $\phi_A$  in adapters and  $\phi_O$  outside the backbone. Thus, the loss and optimization here can be written as follows:

$$\mathcal{L}(\mathcal{D}, \phi_F, \phi_A, \phi_O) = \sum_{i=1}^N \text{loss}(f_{\phi_F, \phi_A, \phi_O}(x_i), y_i), \quad (3)$$

$$\Omega \leftarrow \arg \min_{\Omega} \mathcal{L}(\mathcal{D}, \phi_F, \Omega). \quad (4)$$

The loss and optimization of E<sup>3</sup>VA-tuning can also be represented by equations 3 and 4.

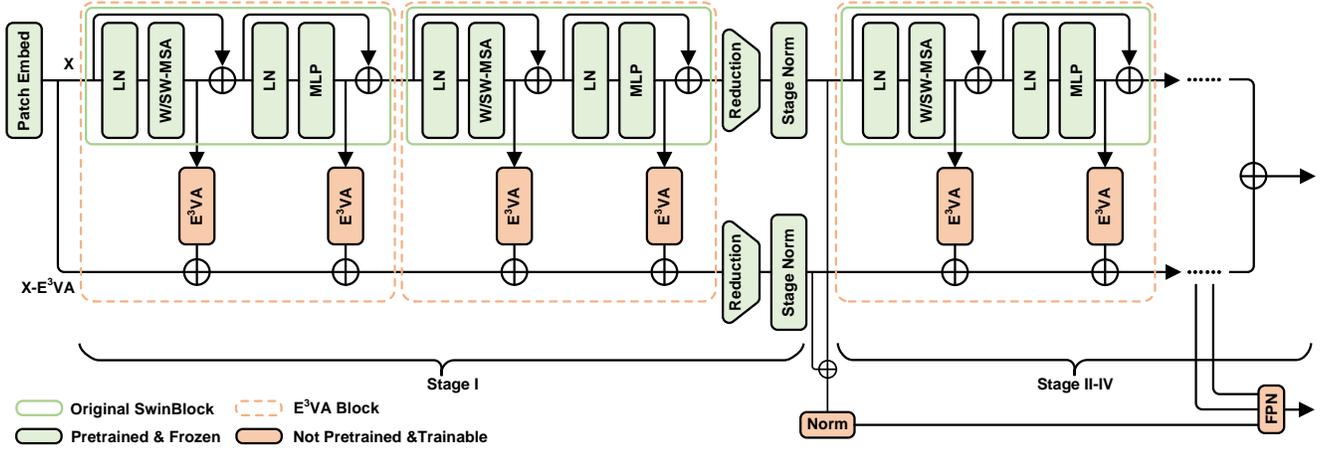
## 3.2 E<sup>3</sup>VA-Tuning

### Gradient Highway

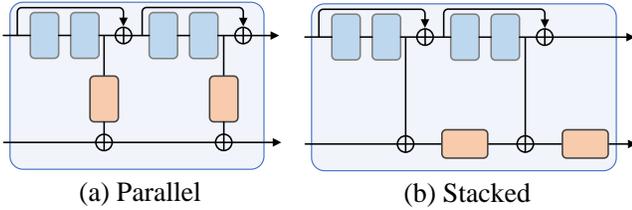
We design a memory and time efficient E<sup>3</sup>VA framework based on Swin Transformer [35]. The overall structure is illustrated in Figure 2. The green box in Figure 2 is the original SwinBlock, and the orange box outlines our E<sup>3</sup>VA Block. Specifically, we separate the E<sup>3</sup>VA adapters from W/SW-MSA and MLP rather than plug them in, thus provide a dedicated "highway" for the adapters. All trainable parameters are on this highway, to calculate gradients for these parameters, the backpropagation only need to go through the adapter highway rather than the frozen layers. In this way, the memory required for massive activations and the computational load during backpropagation are reduced, leading to much reduced training memory and time compared to the full-tuning and other PETL methods. Additionally, we adopt a design of parallel adapters instead of stacked ones, aiming to further shorten the length of the backpropagation path.

### Parallel Adapters

We consider parallel/stacked adapters when designing E<sup>3</sup>VA's highway, as shown in Figure 3. In stacked mode, the output of the first adapter affects the second adapter, while the computations of the two adapters are independent in parallel mode. We adopt parallel adapters design in our E<sup>3</sup>VA as it continuously saves memory and time by simplifying the backpropagation path within a stage compared to the stacked design. Specifically, the output of each adapter is added to the output of previous adapters in this pathway. The summation operations here play two roles: one is to connect all adapters into a link that can be forward propagated, and the other is to imitate the skip-connection operation in Swin-Block. The equations in section 3.4 intuitively describe the forward propagation process of the proposed method. Experiments (see subsequent section 4.3 on ablation experiments) show that parallel designs achieve better performance and faster inference speed. To



**Figure 2: The overall of  $E^3VA$ .** We provide a highway parallel to SwinBlock for the trainable  $E^3VA$  adapters to avoid gradient backpropagation through the backbone. Intermediate activations from W/SA-MSA and MLP are sent to  $E^3VA$ . Downsampling layers are added in the  $E^3VA$  highway and are inherited from the backbone model. The highway is integrated to the FPN with trainable norm .



**Figure 3: Schematic of the stacked/parallel adapters in a SwinBlock.** Parallel adapters are fused by summation, while stacked adapters interact with each other in their propagation.

make  $E^3VA$  applicable in most dense prediction frameworks, we also consider some other details as follows.

#### Downsampling in $E^3VA$

NLP transformers usually have fixed feature dimensions (e.g. T5 [43]). However, CV models downsample multiple times in the backbone, so this difference need to be considered in our  $E^3VA$ . Downsampling in SwinBlock consists of two main layers, a linear layer without bias and a stage norm layer with bias. We insert the same downsampling layers as SwinBlock into the  $E^3VA$  pathway and inherit the corresponding pre-trained parameters directly. Experiments (see section 4.3 on ablation experiments) show that the new trainable downsampling layers introduce numerous additional parameters but cannot bring significant performance gains. Therefore, we directly inherit the pre-trained downsampling layers in the  $E^3VA$  highway and freeze their parameters.

#### FPN in $E^3VA$

The FPN [26] design is an important component of dense prediction models. SwinBlock defines several norm layers before FPN (as shown in Figure 2), but these norm layers are not pre-trained since backbones are pre-trained by classification tasks. In  $E^3VA$ , we integrate the highway into the FPN alongside the backbone and train

the norms before FPN. Experimental results (see the section 4.3 on ablation) show that training these layers can improve performance without additional memory or parameter costs.

### 3.3 Parameter-Efficient $E^3VA$ Adapter

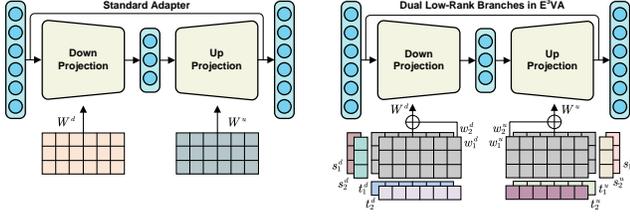
After introducing the framework of  $E^3VA$ , this section describes the parameter-efficient module in  $E^3VA$ -tuning. We will first introduce the standard structure of the adapter and then introduce the  $E^3VA$  adapter.

#### Standard Adapter

The standard adapter structure [14] is illustrated on the left of Figure 4 (bias is hidden). As mentioned in the [23], the computational process for the standard adapter layer can be written as  $A^l = U^l(GeLU(D^l(x))) + x$ , where  $A^l$  is the adapter of layer  $l$ ,  $U^l$  and  $D^l$  denote the up and down projections. Linear layers in up/down projections can be described as:  $y = Wx + b$ . Parameters in linear layers mainly come from  $W$ , which brings lots of parameters especially when the dimension is large.

#### $E^3VA$ Adapter

Karimi *et al.* [23] and He *et al.* [13] demonstrate that low-rank structures can reduce large amounts of parameters in PETL with impressive performance. Therefore, we propose a simple and effective Dual Low-Rank Branches structure to reduce the number of new trainable parameters as much as possible. Our parameter-efficient module is illustrated on the right side of Figure 4. Inspired by the Mixture of Expert (MoE) [38], we approximate a matrix  $W \in \mathbb{R}^{m \times n}$  by the sum of two matrices  $w_1, w_2 \in \mathbb{R}^{m \times n}$  with lower degrees of freedom to increase the robustness of the structure. For each  $w \in \mathbb{R}^{m \times n}$ , we synthesise it by the product of two low-rank matrices  $s \in \mathbb{R}^{m \times \alpha}$  and  $t \in \mathbb{R}^{\alpha \times n}$ . Thus, the up and down projection matrices in  $E^3VA$  can be expressed as  $W^u = \sum_{i=1}^2 w_i^u = \sum_{i=1}^2 s_i^u \times t_i^u$ , and  $W^d = \sum_{i=1}^2 w_i^d = \sum_{i=1}^2 s_i^d \times t_i^d$ .  $W \in \mathbb{R}^{m \times n}$  contains  $mn$  parameter, while  $\hat{W} \in \mathbb{R}^{m \times n}$  synthesized via Dual Low-Rank Branches contains  $2\alpha(m+n)$  parameter. Given  $\alpha \ll \min(m, n)$ , it is obvious

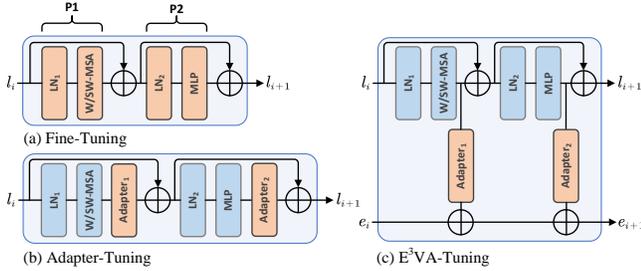


**Figure 4: Internal structure of E<sup>3</sup>VA adapter. Left: Standard adapter structure (bias is hidden). Right: Dual low-rank branches in E<sup>3</sup>VA. Weight  $W$  is the sum of  $w_1$  and  $w_2$  with low degrees of freedom.  $w_1$  and  $w_2$  are synthesized by the product of two low-rank matrices. Low-rank design can reduce most parameters in the adapter and slightly reduce the gradient size of the E<sup>3</sup>VA.**

that  $2\alpha(m+n) \ll mn$ . In Swin Transformer, this design can save over 90% parameters in standard adapters.

### 3.4 Comparisons with adapter-tuning

Here, we illustrate the feature learning difference of E<sup>3</sup>VA with adapter-tuning by analyzing the forward propagation process, and explain why E<sup>3</sup>VA can save a lot of training memory by analyzing the gradient backpropagation process.



**Figure 5: Schematic diagram of the structures of the three paradigms. Orange structures are trainable and blue ones are frozen. (a) In full fine-tuning, all parameters are trained. (b) In adapter-tuning, only the adapters are trained, but the backpropagation go through the backbone. (c) In E<sup>3</sup>VA-tuning, only the adapters are trainable and the backpropagation only need to go through the adapter highway.**

#### Forward Propagation

Figure 5 shows the forward processes of three tuning frameworks in SwinBlock. LN<sub>1</sub> and W/SW-MSA in SwinBlock are simplified as  $f_{P1}$ , LN<sub>2</sub> and MLP are simplified as  $f_{P2}$ . Two adapter layers can be simplified as  $f_{A1}$  and  $f_{A2}$ . For the  $i$ -th block, the output  $l_{i+1}$  of full fine-tuning can be written as follows:

$$l_{i+1} = l_i + f_{P1}(l_i) + f_{P2}(l_i + f_{P1}(l_i)), \quad (5)$$

and that of adapter-tuning is:

$$l_{i+1} = l_i + f_{A1}(f_{P1}(l_i)) + f_{A2}(f_{P2}(l_i + f_{A1}(f_{P1}(l_i)))). \quad (6)$$

E<sup>3</sup>VA has another pair of (input, output),  $(e_i, e_{i+1})$ , and the forward process of  $l$  and  $e$  can be represented as follows:

$$l_{i+1} = l_i + f_{P1}(l_i) + f_{P2}(l_i + f_{P1}(l_i)), \quad (7)$$

$$e_{i+1} = e_i + f_{A1}(f_{P1}(l_i)) + f_{A2}(f_{P2}(l_i + f_{P1}(l_i))). \quad (8)$$

It is easy to see that  $l_{i+1}$  is calculated in the same way in full fine-tuning and E<sup>3</sup>VA-tuning. Equations 6 show that adapters in adapter-tuning are inserted into the backbone so that the subsequent calculation of the backbone depends on the preceding adapters, thus the update of adapters depends on the backpropagation through the backbone. In contrast, the update of E<sup>3</sup>VA's adapters do not depend on the backpropagation of backbone as the adapters are not inserted to the backbone. We rewrite equations 6 and 8 as follows:

$$l_{i+1} - l_i = f_{A1}(f_{P1}(l_i)) + f_{A2}(f_{P2}(l_i + f_{A1}(f_{P1}(l_i)))), \quad (9)$$

$$e_{i+1} - e_i = f_{A1}(f_{P1}(l_i)) + f_{A2}(f_{P2}(l_i + f_{P1}(l_i))). \quad (10)$$

We can see Equations 9 and 10 are very similar except for the last term of adapter-tuning, which has an extra  $f_{A1}$  (i.e.,  $f_{A1}$  is inserted into the backbone). It shows both adapters in adapter-tuning and E<sup>3</sup>VA-tuning take the intermediate activations from the backbone as input. This design keeps E<sup>3</sup>VA-tuning to have sufficient expressiveness as adapter-tuning.

#### Back Propagation

We derive the gradient calculation process for the parameter  $\theta_i$  of the adapter layer in the  $i$ -th block.  $L$  denotes the loss,  $\theta_i = \{\theta_i^1, \theta_i^2, \theta_i^3, \dots, \theta_i^n\}$ , the output of the  $i$ -th layer is  $l_{i+1}$ , and the function of adapter in the  $i$ -th block is  $f_A$ . First, the partial derivative of  $l_{i+1}$  with respect to  $f_A$  is:

$$\frac{\partial l_{i+1}}{\partial f_A} = \frac{\partial l_{i+1}}{\partial f_i^1} \frac{\partial f_i^1}{\partial f_i^2} \frac{\partial f_i^2}{\partial f_i^3} \dots \frac{\partial f_i^m}{\partial f_A}, \quad (11)$$

where  $f_i^j$  is the intermediate process from layer  $l_{i+1}$  to  $f_A$ . Then, the partial derivative of  $L$  with respect to  $\theta_i$  is:

$$\begin{aligned} \frac{\partial L}{\partial \theta_i} &= \frac{\partial L}{\partial l_{i+1}} \frac{\partial l_{i+1}}{\partial f_A} \frac{\partial f_A}{\partial \theta_i} \\ &= \frac{\partial L}{\partial l_{i+1}} \left( \frac{\partial l_{i+1}}{\partial f_i^1} \frac{\partial f_i^1}{\partial f_i^2} \frac{\partial f_i^2}{\partial f_i^3} \dots \frac{\partial f_i^m}{\partial f_A} \right) \sum_{k=1}^n \frac{\partial f_A}{\partial \theta_i^k}. \end{aligned} \quad (12)$$

Standard adapter reduces the number of trainable parameters, but do not reduce the intermediate procedure  $\frac{\partial l_{i+1}}{\partial f_A}$ . Recent PETL methods [4, 5, 13, 20, 34] reduce the number of  $\theta_i^n$  in the adapter, but still do not simplify  $\frac{\partial l_{i+1}}{\partial f_A}$ . In fact, E<sup>3</sup>VA greatly reduces the process of  $\frac{\partial l_{i+1}}{\partial f_A}$  (or  $m$  in equation 11) through a parallel gradient highway, so E<sup>3</sup>VA can save much more memory and time in each block. Moreover, E<sup>3</sup>VA also reduces the processes from loss to the  $i$ -th block  $\frac{\partial L}{\partial l_{i+1}}$ , so the memory advantages are enlarged again.

## 4 Experiments

We conduct comprehensive experiments on mainstream dense prediction tasks to demonstrate the effectiveness and advantages of E<sup>3</sup>VA, including instance segmentation, object detection and semantic segmentation. Experimental settings are introduced in Section 4.1. Main results are presented in Section 4.2. Section 4.3 shows the ablation experiments on several designs. Implementation details and inference comparisons can be found in Appendix.

Swin-L (198M)	Trained* Params	$\Delta_{Full}$	Memory (VOC)	$\Delta_{Full}$ (VOC)	Extra Structure	Pascal VOC (RetinaNet)		ADE20K (UPerNet)	
						AP <sub>Box</sub>	$\Delta_{Full}$	mIoU	$\Delta_{Full}$
<b>Baselines</b>									
FULL	198.58 M	-	15679 MB	-	✗	83.5 %	-	52.10 %	-
FIXED	0.00 M	-100.00 %	3967 MB	-74.70 %	✗	83.6 %	+0.1 %	46.84 %	-5.26 %
BITFIT	0.30 M	-99.85 %	10861 MB	-30.73 %	✗	85.7 %	+2.2 %	48.37 %	-3.73 %
NORM-TUNING	0.09 M	-99.95 %	10123 MB	-35.44 %	✗	85.8 %	+2.3 %	47.98 %	-4.12 %
PARTIAL-1	28.34 M	-85.47 %	3943 MB	-74.85 %	✗	85.4 %	+1.9 %	47.44 %	-4.66 %
ADAPTER	4.66 M	-97.65 %	10793 MB	-31.16 %	✓	87.1 %	+3.6 %	50.78 %	-1.32 %
LoRA	4.57 M	-97.70 %	10127 MB	-35.41 %	✓	<b>87.5 %</b>	<b>+4.0 %</b>	50.34 %	-1.76 %
ADAPTFORMER	4.66 M	-97.65 %	11036 MB	-29.61 %	✓	87.3 %	+3.8 %	50.83 %	-1.27 %
LoRAND	1.31 M	-99.34 %	11986 MB	-23.55 %	✓	86.8 %	+3.3 %	50.76 %	-1.34 %
<b>Our Methods</b>									
E <sup>3</sup> VA	<b>1.79 M</b>	<b>-99.08 %</b>	<b>4819 MB</b>	<b>-69.26 %</b>	✓	86.5 %	+3.0 %	49.64 %	-2.46 %
E <sup>3</sup> VA +	3.53 M	-98.19 %	5175 MB	-66.99 %	✓	86.8 %	+3.3 %	50.20 %	-1.90 %
E <sup>3</sup> VA ++	7.00 M	-96.42 %	5405 MB	-65.53 %	✓	87.0 %	+3.5 %	<b>51.01 %</b>	<b>-1.09 %</b>

**Table 1: Results of baselines and our methods on Pascal VOC and ADE20K datasets. Swin-L is employed as the pre-trained model here.**

Swin-B (87M)	Trained* Params	$\Delta_{Full}$	Memory	$\Delta_{Full}$	Extra Structure	COCO (Cascade Mask R-CNN)			
						AP <sub>Box</sub>	$\Delta_{Full}$	AP <sub>Mask</sub>	$\Delta_{Full}$
<b>Baselines</b>									
FULL	86.75 M	-	17061 MB	-	✗	51.9 %	-	45.0 %	-
FIXED	0.00 M	-100.00 %	7137 MB	-58.17 %	✗	43.5 %	-8.4 %	38.6 %	-6.4 %
BITFIT	0.20 M	-99.77 %	13657 MB	-19.95 %	✗	47.9 %	-4.0 %	41.9 %	-3.1 %
NORM-TUNING	0.06 M	-99.93 %	12831 MB	-24.79 %	✗	48.0 %	-3.9 %	41.4 %	-3.6 %
PARTIAL-1	12.60 M	-85.47 %	7301 MB	-57.21 %	✗	49.2 %	-2.7 %	42.8 %	-2.2 %
ADAPTER	3.11 M	-96.41 %	12557 MB	-26.40 %	✓	50.9 %	-1.0 %	43.8 %	-1.2 %
LoRA	3.03 M	-96.51 %	11975 MB	-29.81 %	✓	51.2 %	-0.7 %	44.3 %	-0.7 %
ADAPTFORMER	3.11 M	-96.41 %	13186 MB	-22.71 %	✓	51.4 %	-0.5 %	44.5 %	-0.5 %
LoRAND	1.20 M	-98.66 %	14038 MB	-17.72 %	✓	51.0 %	-0.9 %	43.9 %	-1.1 %
<b>Our Methods</b>									
E <sup>3</sup> VA	<b>1.20 M</b>	<b>-98.62 %</b>	<b>7639 MB</b>	<b>-55.23 %</b>	✓	50.5 %	-1.4 %	43.8 %	-1.2 %
E <sup>3</sup> VA +	2.35 M	-97.29 %	7761 MB	-54.51 %	✓	51.0 %	-0.9 %	44.2 %	-0.8 %
E <sup>3</sup> VA ++	4.66 M	-94.63 %	8941 MB	-47.59 %	✓	51.6 %	-0.3 %	44.5 %	-0.5 %
E <sup>3</sup> VA(SWIN-L)	1.80 M	-	9471 MB	-	✓	<b>52.2 %</b>	<b>+0.3 %</b>	<b>45.2 %</b>	<b>+0.2 %</b>

**Table 2: Results of baselines and our methods on COCO benchmark. Swin-B is employed as the pre-trained model here. Given that E<sup>3</sup>VA can train the Swin-L-based instance segmentation model with very little memory, we show its results to demonstrate the superiority of the proposed method. Other Swin-L-based baselines cannot be trained on Tesla V100 (batch size is 2 for each), so they are not shown here.**

## 4.1 Experimental Setup

**Dataset.** We conduct extensive experiments on MS COCO [28], ADE20K [65] and Pascal VOC [11]. MS COCO is a commonly used instance segmentation benchmark, which includes 118k training and 5k validation images. All experiments on COCO dataset employ Cascade Mask RCNN [35] as the framework. ADE20K is a widely used semantic segmentation benchmark, including 20k training and 2k validation images. All experiments on ADE20K employ UPerNet [57] as the framework. For the object detection task, we use Pascal

VOC 0712 with 16k training and 5k validation images. Since VOC 0712 has far fewer samples than the latest CV benchmark, we treat it as a low-resource condition in CV. Low-resource conditions can better reflect the advantages of PETL methods. Experiments on VOC employ RetinaNet [27] as the framework.

**Pretrained Backbones.** We conduct experiments on the advanced Swin-Transformer [35] series. All backbones in this section are pre-trained by ImageNet-22K [9].

**Baselines.** We select two kinds of baselines, a total of 9 methods, based on whether extra structures are introduced in backbone. 1) Without Extra Structures. FULL: all parameters in the backbone are trainable. FIXED: fix pre-trained parameters in Swin and train other parts (neck, head). BITFIT [44]: only biases in backbone are trainable. NORM-TUNING: only norm layers in backbone are trainable. PARTIAL-1: the last SwinBlock is trainable, while the other SwinBlocks are frozen. PARTIAL-1: the last SwinBlock is trainable, while the other SwinBlocks are frozen. 2) With Extra Structures (middle dim of these methods are set to 64). ADAPTER [14]: add standard adapters behind MSA and MLP layers in SwinBlocks. LoRA [15]: add trainable matrices in parallel to weight matrices in MSA/MLP. ADAPTFORMER [6]: add adapters and scales in parallel to MSA/MLP of Swin. LoRAND [61]: add LoRand layers after the MSA/MLP layers of each SwinBlock.

**E<sup>3</sup>VA settings.** We experiment on three kinds of E<sup>3</sup>VA settings. The following three settings differ in the rank  $\alpha$  of low-rank matrices  $s \in \mathbb{R}^{m \times \alpha}$  and  $t \in \mathbb{R}^{\alpha \times n}$  ( $n, m$  represent input and middle dimensions). E<sup>3</sup>VA's middle dims are half of input dims. E<sup>3</sup>VA:  $\alpha = 8$ , E<sup>3</sup>VA+:  $\alpha = 16$ , and E<sup>3</sup>VA++:  $\alpha = 32$ .

## 4.2 Main Results

We compare E<sup>3</sup>VA with the baseline methods in terms of the number of trainable parameters, memory, training time and performance, and **bold** the best PETL results. Table 2 shows the results of COCO, which employ Swin-Base as the backbone. Table 1 shows the results of Pascal VOC and ADE20K, which employs Swin-Large. Table 3 compares the training time of different methods. We analyze the GPU usage for Swin-L on COCO in Tables 4. We can summarise three main conclusions from Tables 2~4:

Time	VOC	% <sub>Full</sub>	ADE20K	% <sub>Full</sub>	COCO	% <sub>Full</sub>
FULL	30h	100.00%	49h	100.00%	81h	100.00%
FIXED	12h	40.00%	30h	61.22%	52h	64.20%
BITFIT	23h	76.67%	43h	87.76%	71h	87.65%
NORM-TUNING	23h	76.67%	39h	79.59%	69h	85.19%
PARTIAL-1	14h	46.67%	31h	63.27%	54h	66.67%
ADAPTER	23h	76.67%	43h	87.76%	76h	93.83%
LoRA	23h	76.67%	41h	83.67%	75h	92.60%
ADAPTFORMER	23h	76.67%	43h	87.76%	76h	93.83%
LoRAND	24h	80.00%	46h	93.88%	79h	97.53%
<b>E<sup>3</sup>VA</b>	<b>17h</b>	<b>56.67%</b>	<b>39h</b>	<b>79.59%</b>	<b>69h</b>	<b>85.19%</b>
E <sup>3</sup> VA+	17h	56.67%	39h	79.59%	70h	86.42%
E <sup>3</sup> VA++	17h	56.67%	39h	79.59%	71h	87.65%

Table 3: Time comparisons on three benchmarks.

1) **E<sup>3</sup>VA can save lots of parameters, memory and time.** Tables 2 and 1 show that E<sup>3</sup>VA can save up to 55.2% and 69.3% memory on COCO and VOC, and save 98.6% and 99.1% trainable parameters on COCO and VOC/ADE20K compared to full fine-tuning. Table 3 shows that E<sup>3</sup>VA can save 43.3%, 20.4% and 14.8% time on VOC, ADE20K and COCO respectively compared to full fine-tuning. Compared to PETL methods with extra structures on similar parameter size, E<sup>3</sup>VAs also save much more memory and time. Methods without additional structures (e.g., FIXED and PARTIAL-1) are efficient but cannot achieve competitive performance. Experiments are conducted with batch size 2. If a larger batch size is used, E<sup>3</sup>VA can

Method	Batch Size per GPU	1080Ti 11GB	P100 16GB	3090 24GB	V100 32GB
FULL	1	✗	✗	✓	✓
	2	✗	✗	✗	✗
BITFIT	1	✗	✗	✓	✓
	2	✗	✗	✗	✗
ADAPTER	1	✗	✗	✓	✓
	2	✗	✗	✗	✗
LoRA	1	✗	✗	✓	✓
	2	✗	✗	✗	✗
E <sup>3</sup> VA	1	✓	✓	✓	✓
	2	✗	✓	✓	✓

Table 4: GPU for Swin-L+Cascade Mask RCNN. FULL and other baselines need to be trained on 3090/V100, whereas the proposed method only needs to be trained on 1080Ti/P100.

save more memory, as activations might consume more memory. We can see E<sup>3</sup>VA significantly improves training efficiency.

To illustrate the significant advantage of E<sup>3</sup>VA in situations with limited GPU resources, we present the trainability of the Cascade Mask RCNN (Swin-L-based) with multiple training methods on different types of GPUs in Table 4. For batchsize=1, E<sup>3</sup>VA reduces the minimum training unit from RTX 3090 to GTX 1080Ti. It is worth noting that, for batchsize=2, E<sup>3</sup>VA can train on 16GB Tesla P100 while other methods cannot train even on the expensive 32GB Tesla V100. E<sup>3</sup>VA can increase the batch size per GPU of the same type, thereby reducing the need for multiple GPUs, resulting in even greater performance and efficiency.

2) **E<sup>3</sup>VA achieves comparable performance compared to full fine-tuning and performs better in low-memory regime.** Based on tables 2 and 1, compared to full fine-tuning, E<sup>3</sup>VA series achieves competitive performance on COCO and ADE20K and outperforms full fine-tuning on Pascal VOC. Experiments on E<sup>3</sup>VA variants show that larger ranks bring better results, and E<sup>3</sup>VA++ achieves the best results. Besides, E<sup>3</sup>VA++ outperforms other PETL methods on COCO/ADE20K and performs comparably on VOC. It is worth noting that, with a much smaller memory usage, E<sup>3</sup>VA can tune larger models (e.g., Swin-L+Cascade Mask RCNN) and surpass both full fine-tuning and other PETL methods that utilize the same GPU resources. Certainly, larger models also lead to more inference costs.

3) **E<sup>3</sup>VA can alleviate over-fitting issue in low-resource data.** The Pascal VOC dataset can be treated as a relatively low-resource dataset here. FULL on VOC performs worse than all other PETL methods, even when compared to FIXED. The results demonstrate that full fine-tuning large CV models with insufficient data can result in severe over-fitting, which is similar to findings in NLP. In contrast, PETL methods preserve the powerful knowledge of pre-trained models by freezing the pre-trained parameters. Experiments on VOC show that the E<sup>3</sup>VA series can effectively avoid over-fitting when training on low-resource datasets and achieve promising results.

## 4.3 Ablation Study

**Low-rank/Standard Adapters.** The low-rank structure is introduced to E<sup>3</sup>VA to enhance tuning efficiency. We compare the dual

low-rank adapters with the standard adapters [14] in Table 5 (rows 1 to 3). The intermediate dimension of the standard adapter is set to 32 to ensure similar parameter sizes. It shows that the low-rank setting performs better even with fewer parameters.

Adapter	Parameter Size	AP <sub>box</sub>
E <sup>3</sup> VA Standard	2.35 M	85.4%
E <sup>3</sup> VA Low-rank	1.79 M	86.5%
Structure	Inference Speed	AP <sub>box</sub>
E <sup>3</sup> VA-Stacked	8.2 batch/s	85.9%
E <sup>3</sup> VA-Parallel	8.5 batch/s	86.5%

**Table 5: Ablations on Standard/Low-rank adapters and Parallel/Stacked structures (Pascal VOC).**

**Parallel/Stacked Structures.** We compare the performance and inference speed of parallel and stacked adapters in Table 5 (rows 4 to 6). Parallel structures with fewer computational steps have a notable advantage in inference speed. In addition, Table 5 (rows 4 to 6) shows that parallel adapters also achieve better performance.

Reduction (Swin-L)	COCO (AP <sub>box</sub> )	ADE20K (mIoU)	Pascal VOC (AP <sub>box</sub> )
Inherited	52.2 %	49.64 %	86.5 %
Trainable	51.7 %	49.97 %	86.7 %
FPN Norm (Swin-L)	COCO (AP <sub>box</sub> )	ADE20K (mIoU)	Pascal VOC (AP <sub>box</sub> )
Fixed	51.8 %	48.51 %	85.9 %
Trainable	52.2 %	49.64 %	86.5 %

**Table 6: Ablations on two E<sup>3</sup>VA designs.**

**Ablation study on downsampling and FPN designs.** We also ablate two designs of E<sup>3</sup>VA-tuning (as mentioned in Sec. 3.2) in Table 6. We first test the impact of trainable/inherited stage reduction on the overall performance (“inherited” means the parameters in the newly added downsampling layers are initialised by the pre-trained reduction layers from the backbone and fixed during training). Rows 1 to 3 of Table 6 show that trainable stage reduction only slightly improves performance on VOC and ADE20K, and performs even worse on COCO. Furthermore, the parameter-inefficient trainable stage reduction results in an additional 5~10% new parameters and more memory. So we inherit the pre-trained reduction layers in the original backbone models in our design. Additionally, we compare the trainable/fixed norms before the FPN. Rows 4 to 6 of Table 6 show that the trainable norms significantly improve performance. As a result, we set the norms before FPN trainable which doesn’t bring much extra memory and time costs during the training process.

#### 4.4 Conclusion

Traditional model training paradigms may no longer be suitable and effective for training large models with limited resources. This paper presents a parameter, memory, and time efficient visual adapter

(E<sup>3</sup>VA) tuning for dense predictions, which significantly reduces the memory and time cost of PETL methods in computer vision with promising performance. E<sup>3</sup>VA enhances the possibilities of training larger models for users with insufficient hardware resources, and we hope that E<sup>3</sup>VA can motivate more effective computer vision training paradigms in the era of large models.

#### References

- [1] Armen Aghajanyan, Sonal Gupta, and Luke Zettlemoyer. 2021. Intrinsic Dimensionality Explains the Effectiveness of Language Model Fine-Tuning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. 7319–7328.
- [2] Caisse Amisse, Mario Ernesto Jijón-Palma, and Jorge Antonio Silva Centeno. 2021. Fine-tuning deep learning models for pedestrian detection. *Boletim de Ciências Geodésicas* 27 (2021).
- [3] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems* 33 (2020), 1877–1901.
- [4] Hao Chen, Ran Tao, Han Zhang, Yidong Wang, Wei Ye, Jindong Wang, Guosheng Hu, and Marios Savvides. 2022. Conv-Adapter: Exploring Parameter Efficient Transfer Learning for ConvNets. *arXiv preprint arXiv:2208.07463* (2022).
- [5] Shoufa Chen, GE Chongjian, Zhan Tong, Jiangliu Wang, Yibing Song, Jue Wang, and Ping Luo. [n. d.]. AdaptFormer: Adapting Vision Transformers for Scalable Visual Recognition. In *Advances in Neural Information Processing Systems*.
- [6] Shoufa Chen, Chongjian Ge, Zhan Tong, Jiangliu Wang, Yibing Song, Jue Wang, and Ping Luo. 2022. Adaptformer: Adapting vision transformers for scalable visual recognition. *arXiv preprint arXiv:2205.13535* (2022).
- [7] Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. 2016. Training deep nets with sublinear memory cost. CoRR abs/1604.06174 (2016). *arXiv preprint arXiv:1604.06174* (2016).
- [8] Zhe Chen, Yuchen Duan, Wenhai Wang, Junjun He, Tong Lu, Jifeng Dai, and Yu Qiao. 2022. Vision transformer adapter for dense predictions. *arXiv preprint arXiv:2205.08534* (2022).
- [9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 248–255.
- [10] Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah Smith. 2020. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping. *arXiv preprint arXiv:2002.06305* (2020).
- [11] Mark Everingham, SM Eslami, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. 2015. The pascal visual object classes challenge: A retrospective. *International journal of computer vision* 111, 1 (2015), 98–136.
- [12] Yuxin Fang, Wen Wang, Binhui Xie, Quan Sun, Ledell Wu, Xinggang Wang, Tiejun Huang, Xinlong Wang, and Yue Cao. 2022. Eva: Exploring the limits of masked visual representation learning at scale. *arXiv preprint arXiv:2211.07636* (2022).
- [13] Xuehai He, Chunyuan Li, Pengchuan Zhang, Jianwei Yang, and Xin Eric Wang. 2022. Parameter-efficient fine-tuning for vision transformers. *arXiv preprint arXiv:2203.16329* (2022).
- [14] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *International Conference on Machine Learning*. PMLR, 2790–2799.
- [15] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685* (2021).
- [16] Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. [n. d.]. LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations*.
- [17] Lei Yu, Hongfeng Yu, Wanxuan Lu, Dongshuo Yin, Xian Sun, and Kun Fu. 2024. Airs: Adapter in remote sensing for parameter-efficient transfer learning. *IEEE Transactions on Geoscience and Remote Sensing* (2024).
- [18] Fatsuma Jauro, Haruna Chiroma, Abdulsalam Y Gital, Mubarak Almutairi, M Abdulhamid Shafi’i, and Jamal H Abawajy. 2020. Deep learning architectures in emerging cloud computing architectures: Recent development, challenges and next research trend. *Applied Soft Computing* 96 (2020), 106582.
- [19] Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge Belongie, Bharath Hariharan, and Ser-Nam Lim. 2022. Visual prompt tuning. In *Computer Vision—ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXIII*. Springer, 709–727.
- [20] Shibo Jie and Zhi-Hong Deng. 2022. Convolutional bypasses are better vision transformer adapters. *arXiv preprint arXiv:2207.07039* (2022).

- [21] Chen Ju, Tengda Han, Kunhao Zheng, Ya Zhang, and Weidi Xie. 2022. Prompting visual-language models for efficient video understanding. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXV*. Springer, 105–124.
- [22] Christoph Käding, Erik Rodner, Alexander Freytag, and Joachim Denzler. 2016. Fine-tuning deep neural networks in continuous learning scenarios. In *Asian Conference on Computer Vision*. Springer, 588–605.
- [23] Rabeeh Karimi Mahabadi, James Henderson, and Sebastian Ruder. 2021. Compact: Efficient low-rank hypercomplex adapter layers. *Advances in Neural Information Processing Systems* 34 (2021), 1022–1035.
- [24] Aodi Li, Liansheng Zhuang, Shuo Fan, and Shafei Wang. 2022. Learning Common and Specific Visual Prompts for Domain Generalization. In *Proceedings of the Asian Conference on Computer Vision*. 4260–4275.
- [25] Dongze Lian, Daquan Zhou, Jiashi Feng, and Xinchao Wang. 2022. Scaling & shifting your features: A new baseline for efficient model tuning. *Advances in Neural Information Processing Systems* 35 (2022), 109–123.
- [26] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. 2017. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2117–2125.
- [27] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2017. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*. 2980–2988.
- [28] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*. Springer, 740–755.
- [29] Lingbo Liu, Bruce XB Yu, Jianlong Chang, Qi Tian, and Chang-Wen Chen. 2022. Prompt-matched semantic segmentation. *arXiv preprint arXiv:2208.10159* (2022).
- [30] Nayu Liu, Xian Sun, Hongfeng Yu, Wenkai Zhang, and Guangluan Xu. 2020. Multistage fusion with forget gate for multimodal summarization in open-domain videos. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 1834–1845.
- [31] Nayu Liu, Kaiwen Wei, Xian Sun, Hongfeng Yu, Fanglong Yao, Li Jin, Guo Zhi, and Guangluan Xu. 2022. Assist non-native viewers: Multimodal cross-lingual summarization for how2 videos. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. 6959–6969.
- [32] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *arXiv preprint arXiv:2107.13586* (2021).
- [33] Yitao Liu, Chenxin An, and Xipeng Qiu. 2022. Y-Tuning: An Efficient Tuning Paradigm for Large-Scale Pre-Trained Models via Label Representation Learning. *arXiv preprint arXiv:2202.09817* (2022).
- [34] Yen-Cheng Liu, Chih-Yao Ma, Junjiao Tian, Zijian He, and Zsolt Kira. [n. d.]. Polyhistor: Parameter-Efficient Multi-Task Adaptation for Dense Vision Tasks. In *Advances in Neural Information Processing Systems*.
- [35] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. 2021. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 10012–10022.
- [36] Gen Luo, Minglang Huang, Yiyi Zhou, Xiaoshuai Sun, Guannan Jiang, Zhiyu Wang, and Rongrong Ji. 2023. Towards efficient visual adaption via structural re-parameterization. *arXiv preprint arXiv:2302.08106* (2023).
- [37] Gen Luo, Yiyi Zhou, Tianhe Ren, Shengxin Chen, Xiaoshuai Sun, and Rongrong Ji. 2024. Cheap and quick: Efficient vision-language instruction tuning for large language models. *Advances in Neural Information Processing Systems* 36 (2024).
- [38] Jiaqi Ma, Zhe Zhao, Xinyang Yi, Jilin Chen, Lichan Hong, and Ed H Chi. 2018. Modeling task relationships in multi-task learning with multi-gate mixture-of-experts. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 1930–1939.
- [39] Xing Nie, Bolin Ni, Jianlong Chang, Gaomeng Meng, Chunlei Huo, Zhaoxiang Zhang, Shiming Xiang, Qi Tian, and Chunhong Pan. 2022. Pro-tuning: Unified prompt tuning for vision tasks. *arXiv preprint arXiv:2207.14381* (2022).
- [40] Junting Pan, Ziyi Lin, Xiatian Zhu, Jing Shao, and Hongsheng Li. 2022. ST-Adapter: Parameter-Efficient Image-to-Video Transfer Learning. *Advances in Neural Information Processing Systems* 35 (2022), 26462–26477.
- [41] Matthew E Peters, Sebastian Ruder, and Noah A Smith. 2019. To tune or not to tune? adapting pretrained representations to diverse tasks. *arXiv preprint arXiv:1903.05987* (2019).
- [42] Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2021. AdapterFusion: Non-destructive task composition for transfer learning. In *16th Conference of the European Chapter of the Association for Computational Linguistics, EAACL 2021*. Association for Computational Linguistics (ACL), 487–503.
- [43] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.* 21, 140 (2020), 1–67.
- [44] Shauli Ravfogel, Elad Ben-Zaken, and Yoav Goldberg. 2021. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked languagemodels. *arXiv preprint arXiv:2106.10199* (2021).
- [45] Andreas Rücklé, Gregor Geigle, Max Glockner, Tilman Beck, Jonas Pfeiffer, Nils Reimers, and Iryna Gurevych. 2021. AdapterDrop: On the Efficiency of Adapters in Transformers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. 7930–7946.
- [46] Chompunuch Sarasaen, Soumick Chatterjee, Mario Breitkopf, Georg Rose, Andreas Nürnberger, and Oliver Speck. 2021. Fine-tuning deep learning model parameters for improved super-resolution of dynamic mri with prior-knowledge. *Artificial Intelligence in Medicine* 121 (2021), 102196.
- [47] Shaden Smith, Mostofa Patwary, Brandon Norick, Patrick LeGresley, Samyam Rajbhandari, Jared Casper, Zhun Liu, Shrimai Prabhumoye, George Zerveas, Vijay Korthikanti, et al. 2022. Using deepspeed and megatron to train megatron-turing nlg 530b, a large-scale generative language model. *arXiv preprint arXiv:2201.11990* (2022).
- [48] Kihyuk Sohn, Yuan Hao, José Lezama, Luisa Polania, Huiwen Chang, Han Zhang, Irfan Essa, and Lu Jiang. 2022. Visual Prompt Tuning for Generative Transfer Learning. *arXiv preprint arXiv:2210.00990* (2022).
- [49] Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and Policy Considerations for Deep Learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 3645–3650.
- [50] Xian Sun, Dongshuo Yin, Fei Qin, Hongfeng Yu, Wanxuan Lu, Fanglong Yao, Qibin He, Xingliang Huang, Zhiyuan Yan, Peijin Wang, et al. 2023. Revealing influencing factors on global waste distribution via deep-learning based dumpsite detection from satellite imagery. *Nature Communications* 14, 1 (2023), 1444.
- [51] Yi-Lin Sung, Jaemin Cho, and Mohit Bansal. 2022. Lst: Ladder side-tuning for parameter and memory efficient transfer learning. *arXiv preprint arXiv:2206.06522* (2022).
- [52] Edna Chebet Too, Li Yujian, Sam Njuki, and Liu Yingchun. 2019. A comparative study of fine-tuning deep learning models for plant disease identification. *Computers and Electronics in Agriculture* 161 (2019), 272–279.
- [53] Haixin Wang, Jianlong Chang, Xiao Luo, Jinan Sun, Zhouchen Lin, and Qi Tian. 2023. LION: Implicit Vision Prompt Tuning. *arXiv preprint arXiv:2303.09992* (2023).
- [54] Wenhui Wang, Hangbo Bao, Li Dong, Johan Bjorck, Zhiliang Peng, Qiang Liu, Kriti Aggarwal, Owais Khan Mohammed, Saksham Singhal, Subhojit Som, et al. 2022. Image as a foreign language: Beit pretraining for all vision and vision-language tasks. *arXiv preprint arXiv:2208.10442* (2022).
- [55] Wenhui Wang, Jifeng Dai, Zhe Chen, Zhenhang Huang, Zhiqi Li, Xizhou Zhu, XiaoWei Hu, Tong Lu, Lewei Lu, Hongsheng Li, et al. 2022. Internimage: Exploring large-scale vision foundation models with deformable convolutions. *arXiv preprint arXiv:2211.05778* (2022).
- [56] Qiong Wu, Wei Yu, Yiyi Zhou, Shubin Huang, Xiaoshuai Sun, and Rongrong Ji. 2024. Parameter and computation efficient transfer learning for vision-language pre-trained models. *Advances in Neural Information Processing Systems* 36 (2024).
- [57] Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. 2018. Unified perceptual parsing for scene understanding. In *Proceedings of the European conference on computer vision (ECCV)*. 418–434.
- [58] Yi Xin, Junlong Du, Qiang Wang, Zhiwen Lin, and Ke Yan. 2024. Vmt-adapter: Parameter-efficient transfer learning for multi-task dense scene understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 16085–16093.
- [59] Jianwei Yang, Chunyuan Li, Xiyang Dai, and Jianfeng Gao. 2022. Focal modulation networks. *Advances in Neural Information Processing Systems* 35 (2022), 4203–4217.
- [60] Dongshuo Yin, LeiYi Hu, Bin Li, and Youqun Zhang. 2023. Adapter is All You Need for Tuning Visual Tasks. *arXiv e-prints* (2023), arXiv–2311.
- [61] Dongshuo Yin, Yiran Yang, Zhechao Wang, Hongfeng Yu, Kaiwen Wei, and Xian Sun. 2023. 1% VS 100%: Parameter-Efficient Low Rank Adapter for Dense Predictions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 20116–20126.
- [62] Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. 2021. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. *arXiv preprint arXiv:2106.10199* (2021).
- [63] Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. 2022. Scaling vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 12104–12113.
- [64] Aston Zhang, Yi Tay, SHUAI Zhang, Alvin Chan, Anh Tuan Luu, Siu Hui, and Jie Fu. 2020. Beyond Fully-Connected Layers with Quaternions: Parameterization of Hypercomplex Multiplications with  $1/n$  Parameters. In *International Conference on Learning Representations*.
- [65] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. 2017. Scene parsing through ade20k dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 633–641.
- [66] Zhuofan Zong, Guanglu Song, and Yu Liu. 2022. DETRs with Collaborative Hybrid Assignments Training. *arXiv preprint arXiv:2211.12860* (2022).