Parameter-, Memory-, Time-Efficient Multi-Task Dense Vision Adaptation

Haiming Yao [1,3], Wei Luo[1,3], Qiyu Chen[2,3], Jianxing Liao[3], Wei You[3*]

[1]Department of Precision Instruments, Tsinghua University, Beijing
[2]Institute of Automation, Chinese Academy of Sciences, Beijing
[3]Advanced Computing and Storage Lab, Huawei Technologies Co. Ltd
{yhm22, luow23}@mails.tsinghua.edu.cn; chenqiyu2021@ia.ac.cn; {liaojianxing, youwei22}@huawei.com

Abstract

While adapting pretrained vision models to downstream dense prediction tasks is widely used, current methods often overlook adaptation efficiency, especially in the context of multi-task learning (MTL). Although parameter-efficient fine-tuning (PEFT) methods can enhance parameter efficiency, broader aspects such as GPU memory and training time efficiency remain underexplored. In this paper, we propose a new paradigm that simultaneously achieves efficiency in Parameters, GPU Memory, and Training Time for Multi-Task Dense Vision Adaptation. Specifically, we propose a dual-branch framework, in which a frozen pretrained backbone serves as the generic main branch, and the proposed Bi-Directional Task Adaptation (BDTA) modules are integrated in parallel to form a task bypass branch that extracts adaptation features required by multiple specific tasks. This adaptation module is lightweight, efficient, and does not require backpropagation through the large pretrained backbone, thus avoiding resource-intensive gradient computations. Moreover, a Mixture of Task Experts mechanism (MoTE) is further proposed to integrate adaptation features across tasks and scales, thereby obtaining more robust representations tailored for dense prediction tasks. On the PASCAL-Context benchmark, our method achieves over 2× relative performance improvement compared to the best prior multi-task PEFT method, while using only $\sim 30\%$ of the parameters, $\sim 50\%$ of the memory, and $\sim 60\%$ of the training time, demonstrating superior overall adaptation efficiency.

1 Introduction

In recent years, the scale effect (Kaplan et al. 2020) has empowered vision models (Caron et al. 2021) with remarkable transferability and adaptability across a wide range of downstream tasks (Awais et al. 2025), primarily through large-scale pre-training followed by task-specific fine-tuning (Oquab et al. 2023; Li et al. 2023). Traditional adaptation methods (Chen et al. 2023a; Xia et al. 2024) often require full fine-tuning of the entire model for specific local vision tasks, including semantic segmentation (Kirillov et al. 2023), depth estimation (Yang et al. 2024), and related objectives. For users with limited computational resources, training and storing such large models poses a serious challenge. This issue becomes even more pronounced in multi-task learning (MTL) scenario (Xu et al. 2023), where each downstream task typically demands its own customized fine-tuning, thereby further amplifying the total adaptation cost.

In the field of natural language processing (NLP), Large Language Models (LLMs) (Ding et al. 2023; Houlsby et al. 2019) have become increasingly dominant. To enable efficient adaptation of these models, numerous Parameter-Efficient Fine-Tuning (PEFT) methods (Han et al. 2024) have been proposed, allowing pre-trained LLMs to be adapted using only a small number of tunable parameters.

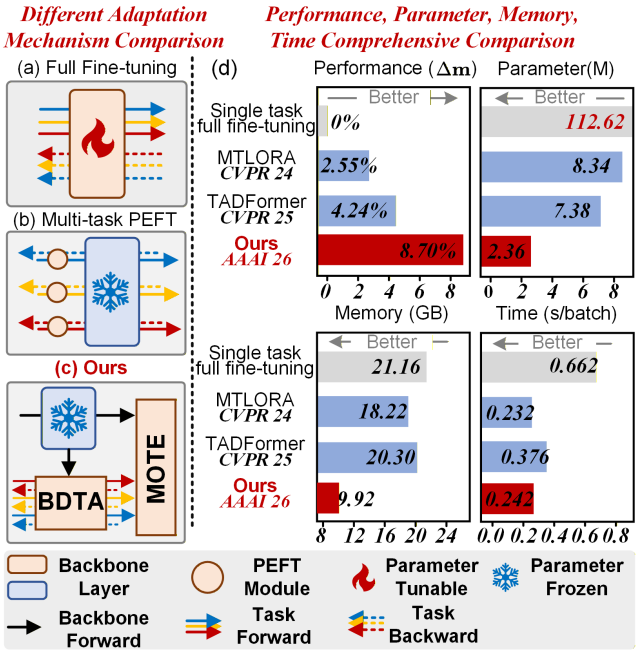

Figure 1: Comparison of different adaptation paradigms for MTL. (a) Full fine-tuning adaptation. (b) Multi-task PEFT methods. (c) The proposed method. (d) Comparison of performance and efficiency on PASCAL-Context among the single-task full fine-tuning baseline and multi-task PEFT methods MTLoRA (Agiza, Neseem, and Reda 2024) and TADFormer (Baek et al. 2025).

*Corresponding author.

These approaches have demonstrated significantly improved efficiency compared to traditional full fine-tuning. Motivated by these successes, PEFT techniques have also been extended to the visual domain (Xin et al. 2024b), with representative examples including adapters (Chen et al. 2022; Jie et al. 2024), low-rank adaptation (LoRA) (Hu et al. 2022b), and visual prompt tuning (Jia et al. 2022; Han et al. 2023). Although these PEFT methods have achieved success compared to full fine-tuning, we observe that they still exhibit unresolved limitations in attaining more ***Comprehensive Efficiency*** under ***Multi-Task Learning*** (MTL) contexts.

MTL seeks to employ a single model to simultaneously perform multiple downstream tasks. In the full fine-tuning paradigm (Fig. 1(a)), task-specific gradients are directly propagated to the backbone, leading to substantial trainable parameter overhead in both optimization and storage. To improve parameter efficiency, recent works have explored applying PEFT to MTL, as shown in Fig. 1(b). For instance, Polyhistor (Liu et al. 2022) and VMT-Adapter (Xin et al. 2024a) extend the adapter architecture to MTL, while MTLoRA (Agiza, Neseem, and Reda 2024) represents a multi-task extension of LoRA. However, we identify a fundamental limitation in these PEFT-based approaches: they prioritize parameter efficiency while overlooking other critical aspects of adaptation efficiency, such as GPU memory usage and training time. This is because from a topological perspective, current PEFT modules (*e.g.*, LoRA and Adapter) are typically inserted sequentially into the pretrained backbone. As a result, although the backbone parameters remain frozen during fine-tuning, gradients must still be computed for the intermediate backbone layers along the path from the prediction output to the PEFT modules. These gradient computations increase GPU memory consumption and training time, thereby reducing overall efficiency.

To address the aforementioned limitations, this paper explores a route toward more comprehensive efficiency in visual adaptation within the MTL context, particularly for challenging dense vision tasks such as semantic segmentation, surface normal estimation, and other pixel-level predictions, by jointly considering parameters, GPU memory usage, and training time. As illustrated in Fig. 1(c), instead of serially inserting adaptation modules into the backbone, our framework adopts a parallel dual-branch architecture to acquire task-specific adaptation features. The pre-trained backbone serves as a generic main branch, while the proposed Bi-Directional Task Adaptation (BDTA) modules are connected in parallel with the backbone, forming a task bypass branch to learn task feature adaptations. This bypass branch directly produces task-specific predictions without interacting with intermediate backbone layers, thereby avoiding complex gradient computations during backpropagation. To enable fine-grained feature adaptation for dense prediction tasks, BDTA modules are integrated at multiple stages of the backbone, allowing for the extraction of hierarchical, multi-scale adaptation features. On top of the adapted features, we introduce a Mixture of Task Experts (MoTE) mechanism, which treats each set of task-specific adapted features as a knowledgeable expert and dynamically fuses them to enable effective adaptation fusion across tasks. Through these designs, our model achieves adaptation tailored for Multi-Task Dense Vision Tasks, while simultaneously achieving **Comprehensive Efficiency** in terms of parameters, GPU memory usage, and training time, as illustrated in Fig. 1(d).

Our main contributions are summarized as follows:

- We identify inefficiencies in existing PEFT methods beyond parameters. To the best of our knowledge, this work presents the first framework that simultaneously considers parameter, memory, and time efficiency for adapting pre-trained vision models to multi-task dense prediction.
- We propose a parallel dual-branch architecture that integrates task bypass branch with generic main branch via the proposed Bi-Directional Task Adaptation (BDTA) modules. Furthermore, the Mixture of Task Experts (MoTE) mechanism is introduced to enable cross-task fusion.
- For multi-task scene understanding on PASCAL-Context, our method surpasses the previous best multi-task PEFT approach, TADFormer (Baek et al. 2025), with over 2× relative performance gain, using only $\sim 30\%$ of the parameters, $\sim 50\%$ of the GPU memory, and $\sim 60\%$ of the training time—demonstrating superior adaptation efficiency (Fig. 1(d)).

2 Related Work

2.1 Efficient Model Adaptation

Efficient model adaptation aims to adapt pre-trained models to new tasks by training only a small subset of parameters, which can be achieved through various routes. The early transfer learning (Niu et al. 2021) involved freezing the parameters of the feature extraction backbone and fine-tuning only the output layer to adapt the model to the target domain. PEFT (Han et al. 2024; Xin et al. 2024b) represents a modern approach that encompasses a range of widely studied paradigms. Adapters (Yi-Lin Sung 2022; Pfeiffer et al. 2020; Mercea et al. 2024) are bottleneck-structured modules that are embedded at various locations within pre-trained models. Low-Rank Adaptation (LoRA) (Hu et al. 2022a) approximates larger tensors using multiple low-dimensional tensors for their updates. Prompt tuning (Zhou et al. 2022; Jia et al. 2022) introduces a small number of learnable prompt tokens to guide the model's adaptation to downstream tasks. While these methods primarily focus on parameter efficiency, recent works such as LST (Sung, Cho, and Bansal 2022), LoSA (Mercea et al. 2024), and E3VA (Yin et al. 2024) have begun to address more comprehensive forms of efficiency, including memory and time. However, these methods are limited to single-task adaptation. To the best of our knowledge, comprehensively efficient adaptation for MTL still remain underexplored.

2.2 Multi-Task Vision Adaptation

To address the needs of model adaptation in MTL scenarios, particularly in the realm of computer vision, several multi-task efficient vision adaptation strategies have recently been explored. Polyhistor (Liu et al. 2022) proposes a cross-task and cross-layer low-rank matrix decomposition technique to achieve parameter sharing within adapters. VMT-Adapter (Xin et al. 2024a) introduces task-shared adapters

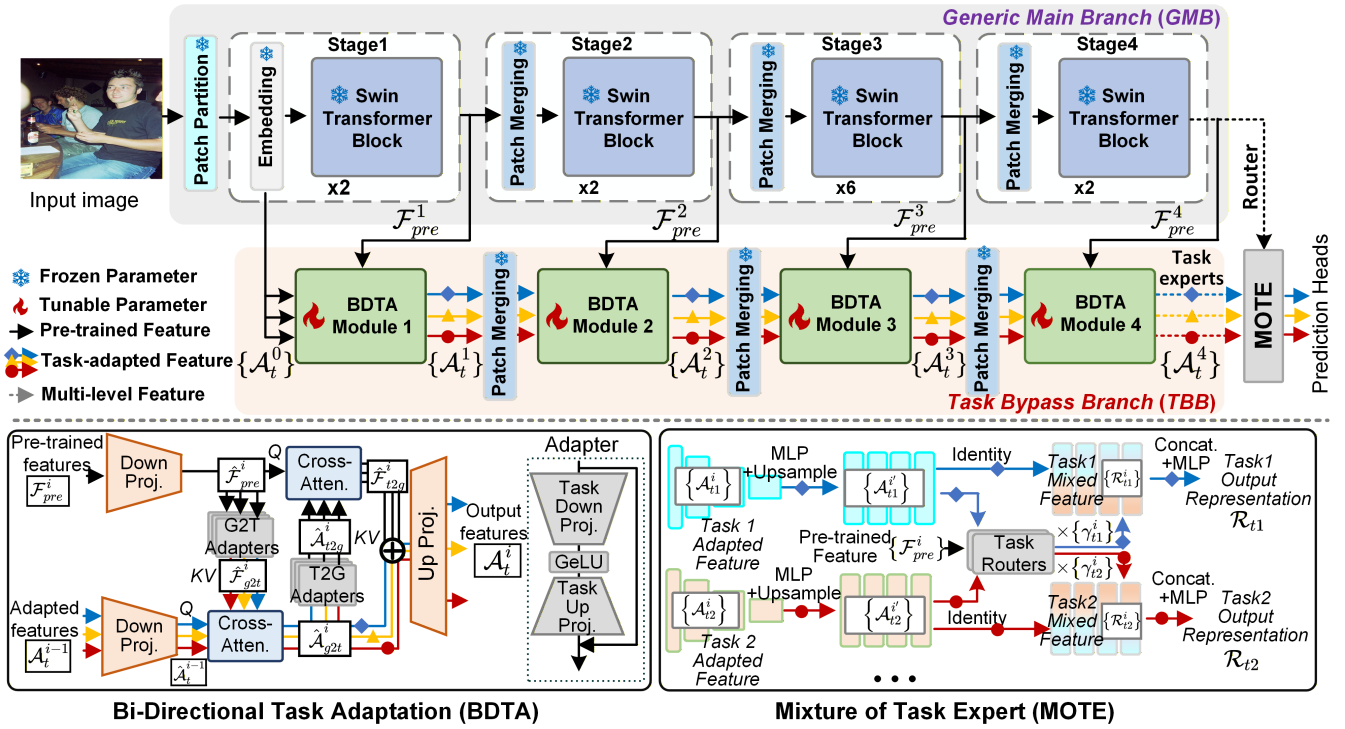

Figure 2: Overall architecture of the proposed multi-task learning framework. The model comprises a frozen generic main branch (GMB) built upon a pre-trained Swin-Tiny (Liu et al. 2021) and a tunable task bypass branch (TBB) with four Bi-Directional Task Adaptation (BDTA) modules that progressively refine task-specific features. A Mixture of Task Experts (MoTE) module integrates multi-level task-adapted features for final task-specific predictions.

and task-specific modules to achieve efficient model execution. MTLoRA (Agiza, Neseem, and Reda 2024) extends LoRA to multi-task adaptation through task-specific low-rank matrices. DiTask (Mantri et al. 2025) achieves task-specific adaptation through the neural diffeomorphic transformation of singular values. TADFormer (Baek et al. 2025) introduces task prompts and dynamic task filters to capture task-related contextual information. However, the adaptation modules introduced in these methods are directly embedded into the pre-trained backbone network, focusing solely on parameter efficiency while neglecting the comprehensive efficiency of GPU memory and training time. In contrast, comprehensive adaptation efficiency is crucial for building resource-friendly multi-task models and warrants further investigation.

3 Methodology

3.1 Dual-Branch Overview

Our proposed method is used to integrate with pre-trained vision models. Fig. 2(a) illustrates the overall architecture of our method when adapted to the hierarchical Swin Transformer model (Liu et al. 2021). The overall framework is a dual-branch structure. First, we regard the pre-trained backbone as the Generic Main Branch (GMB). Given an input image $\mathbf{X}$, the forward propagation through its N stages is expressed as:

$$\begin{aligned} \mathcal{F}_{pre}^{0} &= \text{PatchEmbed}\left(\mathbf{X}\right); \\ \mathcal{F}_{pre}^{i} &= b_i\left(\mathcal{F}_{pre}^{i-1}\right), i = \{1, ..., N\}, \end{aligned} \tag{1}$$

where the b_i is the i-th stage of the GMB, where each stage consists of a patch merging layer and Swin Transformer blocks, and $\mathcal{F}_{pre}^{i} \in \mathbb{R}^{L_i \times C_i}$ denotes the i-th intermediate out with the L_i tokens of dimension C_i. Then, as shown in Fig. 2(a), to adapt the pre-trained features to K downstream tasks $T = \{t_1, t_2, ..., t_K\}$, we also construct the task bypass branch (TBB):

$$\begin{aligned} \mathcal{A}_{t}^{0} &= \text{PatchEmbed}\left(\mathbf{X}\right), t \in T; \\ \mathcal{A}_{t}^{i} &= a_i\left(\mathcal{A}_{t}^{i-1}, \mathcal{F}_{pre}^{i}\right), i = \{1, ..., N\}, \end{aligned} \tag{2}$$

where the a_i is i-th adaptation stage of the TBB, which is composed of a Bi-Directional Task Adaptation (BDTA) module and a patch merging layer inherited from the corresponding stage of the GMB, and $\mathcal{A}_{t}^{i} \in \mathbb{R}^{L_i \times C_i}$ denotes the i-th adaptation out for task t. For the first stage, we directly utilize the patch embeddings from the GMB as the initial feature for all tasks T.

Finally, the obtained feature sets $\left\{\mathcal{F}_{pre}^{i}\right\}, i = \{1, ..., N\}$ from GMB and $\left\{\mathcal{A}_{t}^{i}\right\}, i = \{1, ..., N\}, t \in T$ from TBB are passed to the Mixture of Task Experts (MoTE) module for cross-task adaptation interactions. The final representations are then processed by the prediction heads for final outputs.

3.2 Bi-Directional Task Adaptation

The detailed architecture of the BDTA module is depicted in Fig. 2(b). Specifically, for the i-th BDTA module, it takes the intermediate features $\mathcal{F}^i_{pre}$ from the i-th stage of the GMB, as well as the adapted features $\{\mathcal{A}^{i-1}_t, t \in T\}$ from the preceding TBB stage as inputs. To begin with, we use two projection layers to reduce the hidden dimensions of the inputs from both branches to $\frac{1}{r}$ of their original size, where r is a reduction factor, obtaining the compressed representations $\hat{\mathcal{F}}^i_{pre} \in \mathbb{R}^{L_i \times \frac{C_i}{r}}$ and $\hat{\mathcal{A}}^{i-1}_t \in \mathbb{R}^{L_i \times \frac{C_i}{r}}$.

In our method, we decouple the adaptation features required for different tasks into a combination of general and task-specific ones, where the pre-trained features are regarded as general features across tasks. Accordingly, we propose a bidirectional adaptation mechanism, comprising General-to-Task adaptation ($G2T$) and Task-to-General adaptation ($T2G$). This process can be formulated as:

$$\hat{\mathcal{A}}^i_{g2t} = \mathcal{H}^i_{g2t}(\hat{\mathcal{A}}^{i-1}_t, \hat{\mathcal{F}}^i_{pre}), \hat{\mathcal{F}}^i_{t2g} = \mathcal{H}^i_{t2g}(\hat{\mathcal{F}}^i_{pre}, \hat{\mathcal{A}}^i_{g2t}), \quad (3)$$

where the $\mathcal{H}_i$ is the i-th adaptation mechanism, which is mainly based on adapters and cross-attention and will be elaborated below.

G2T Adaptation. We first introduce $G2T$ adaptation $\mathcal{H}^i_{g2t}$. Specifically, the $G2T$ adapters of each task t are employed to adapt general features into task-specific features:

$$\hat{\mathcal{F}}^i_{g2t} = \mathrm{GeLU}(\hat{\mathcal{F}}^i_{pre} \cdot W^{g2t}_{down}) \cdot W^{g2t}_{up} + \hat{\mathcal{F}}^i_{pre}, t \in T, \quad (4)$$

where $\hat{\mathcal{F}}^i_{g2t}$ is the adapted feature for task t in i-th stage, and W^{g2t}_{down}, W^{g2t}_{up} are the projection weights of the adapters. Subsequently, we treat $\hat{\mathcal{F}}^i_{g2t}$ as key-value pairs and $\hat{\mathcal{A}}^{i-1}_t$ as queries, and inject $\hat{\mathcal{F}}^i_{g2t}$ into $\hat{\mathcal{A}}^{i-1}_t$ using cross-attention followed by a feed-forward network (FFN):

$$\begin{aligned} \tilde{\mathcal{A}}^{i-1}_{g2t} &= \hat{\mathcal{A}}^{i-1}_t + \mathrm{Attention}(\mathrm{LN}(\hat{\mathcal{A}}^{i-1}_t), \mathrm{LN}(\hat{\mathcal{F}}^i_{g2t})), \\ \hat{\mathcal{A}}^i_{g2t} &= \tilde{\mathcal{A}}^{i-1}_{g2t} + \mathrm{FFN}(\mathrm{LN}(\tilde{\mathcal{A}}^{i-1}_{g2t})), \end{aligned} \quad (5)$$

where the $\mathrm{LN}(\cdot)$ is the LayerNorm, for the attention operation $\mathrm{Attention}(\cdot)$, we use sparse attention (Chen et al. 2023b) to reduce the computational complexity. The generated feature $\hat{\mathcal{A}}^i_{g2t}$ will be used as the input of the $T2G$ adaptation $\mathcal{H}^i_{t2g}$.

T2G Adaptation. We construct the $T2G$ adaptation $\mathcal{H}^i_{t2g}$ to extract task-general representations from each $\hat{\mathcal{A}}^i_{g2t}$ and incorporate them into $\hat{\mathcal{F}}^i_{pre}$. Similarly, the $T2G$ adapters with projection weights of W^{t2g}_{down}, W^{t2g}_{up} of each task t are used to adapt task-specific features into general features $\hat{\mathcal{A}}^i_{t2g}$:

$$\hat{\mathcal{A}}^i_{t2g} = \mathrm{GeLU}(\hat{\mathcal{A}}^i_{g2t} \cdot W^{t2g}_{down}) \cdot W^{t2g}_{up} + \hat{\mathcal{A}}^i_{g2t}, t \in T. \quad (6)$$

Then, $\hat{\mathcal{A}}^i_{t2g}$ are injected into $\hat{\mathcal{F}}^i_{pre}$ as the following process:

$$\begin{aligned} \tilde{\mathcal{F}}^i_{t2g} &= \hat{\mathcal{F}}^i_{pre} + \mathrm{Attention}(\mathrm{LN}(\hat{\mathcal{F}}^i_{pre}), \mathrm{LN}(\hat{\mathcal{A}}^i_{t2g})), \\ \hat{\mathcal{F}}^i_{t2g} &= \tilde{\mathcal{F}}^i_{t2g} + \mathrm{FFN}(\mathrm{LN}(\tilde{\mathcal{F}}^i_{t2g})), \end{aligned} \quad (7)$$

in which we use the $\hat{\mathcal{F}}^i_{pre}$ as the query, and the adapted feature $\hat{\mathcal{A}}^i_{t2g}$ as the key-value pairs. After the aforementioned bi-directional adaptation process, the general and task-specific features have been fully interacted. We take their sum as the features required for task adaptation:

$$\hat{\mathcal{A}}^i_t = \hat{\mathcal{A}}^i_{g2t} + \hat{\mathcal{F}}^i_{t2g}, t \in T. \quad (8)$$

Finally, an output up-projection layer is employed to map the hidden dimensions of $\hat{\mathcal{A}}^i_t$ back to the original size C_i as $\mathcal{A}^i_t$, which will then be fed into the next BDTA module.

3.3 Mixture of Task Experts

After the aforementioned adaptation process, each feature set $\left\{\mathcal{A}^i_t\right\}, i = \{1, ..., N\}, t \in T$ captures the representative biases of each task, which we treat as the task-specific experts. To facilitate cross-task interactions, we propose the Mixture of Task Experts (MoTE) module, as illustrated in Fig. 2(c). First, inspired by SegFormer (Xie et al. 2021), we employ lightweight task-specific MLP layers $W^i_t \in \mathbb{R}^{C_i \times C_1}, i = \{1, ..., N\}, t \in T$ to reduce the channel dimensions of the multi-stage features $\left\{\mathcal{A}^i_t\right\}$ and up-sample their spatial sizes as:

$$\mathcal{A}^{i'}_t = \mathrm{Upsample}(\mathcal{A}^i_t \cdot W^i_t) \in \mathbb{R}^{L_1 \times C_1}, i = \{1, ..., N\}, t \in T. \quad (9)$$

We then design task routers to generate gating scores for task expert representations $\{\mathcal{A}^{i'}_t\}$. Specifically, for i-th stage and task t, the task router r^i_t takes the pre-trained feature $\mathcal{F}^i_{pre} \in \mathbb{R}^{L_i \times C_i}$ as input, and employs a two-layer MLP to map its channels to the number of downstream tasks K and perform global pooling to obtain a feature vector $h^i_t \in \mathbb{R}^K$. We further introduce a noise gating mechanism (Shazeer et al. 2017) by generating a noise vector $e^i_t \in \mathbb{R}^K$ through another two-layer MLP, and the final gating score $g^i_t \in \mathbb{R}^K$ can be obtained as:

$$g^i_t = \mathrm{Softmax}\left(h^i_t + \mathcal{N}(0,1)\,\mathrm{Softplus}\left(e^i_t\right)\right). \quad (10)$$

The obtained gating scores are used as the weights to fuse the task expert representations, and we define the cross-task mixed representations $\mathcal{R}^i_t$ as:

$$\mathcal{R}^i_t = \mathcal{A}^{i'}_t + \gamma^i_t \odot \textstyle\sum_{q \in T} g^i_t(q)\, \mathcal{A}^{i'}_q, t \in T, i = \{1, ..., N\}, \quad (11)$$

where $\odot$ represents the Hadamard product, $g^i_t(q)$ denotes the entry of g^i_t corresponding to task q, and $\gamma^i_t \in \mathbb{R}^{C_1}$ is a learnable vector to balance the cross-task fused representation and the task expert representation, which is initialized to zero. This initialization strategy ensures that $\mathcal{A}^{i'}_t$ can learn task-specific basis representations for each task t without being disrupted by cross-task fused representations. Finally, MLP layers $W_t \in \mathbb{R}^{NC_1 \times C_1}, t \in T$ are adopted to fuse the concatenated features of different stages $\{\mathcal{R}^i_t\}$ for each task t to obtain the output representation as $\mathcal{R}_t = \left(\mathrm{Concat}\left(\mathcal{R}^i_t\right), i = \{1, ..., N\}\right) \cdot W_t$, which are fed into prediction heads to generate the task prediction outputs.

3.4 Efficiency Analysis

We evaluate the efficiency of our method from three perspectives: Parameters, GPU memory, and Training time.

In terms of parameters, our approach introduces only a small number of trainable parameters. Each BDTA module

Method	Taxonomy	SemSeg (mIoU ↑)	HumPa (mIoU ↑)	Saliency (mIoU ↑)	Normals (RMSE ↓)	$\Delta\mathbf{m}(\%)\uparrow$	#TP(M)↓
TRADITIONAL FINE-TUNING METHODS							
Single-task full fine-tuning	–	67.21	61.93	62.35	17.97	0	112.62
Multi-task full fine-tuning	–	68.71	**62.13**	64.18	17.35	+2.23	30.06
Decoder-only fine-tuning	–	63.14	52.37	58.39	20.89	-11.02	2.55
SINGLE-TASK PEFT-BASED METHODS							
Adapter (He et al. 2022)	*ICLR 22*	69.21	57.38	61.28	18.83	-2.71	11.24
VL-Adapter (Yi-Lin Sung 2022)	*CVPR 22*	70.21	59.15	62.29	19.26	-1.83	4.74
LoRA (Hu et al. 2022a)	*ICLR 22*	70.12	57.73	61.90	18.96	-2.17	2.87
VPT-shallow (Jia et al. 2022)	*ECCV 22*	62.96	52.27	58.31	20.90	-11.18	2.57
VPT-deep (Jia et al. 2022)	*ECCV 22*	64.35	52.54	58.15	21.07	-10.85	3.43
BitFit (Zaken, Goldberg, and Ravfogel 2022)	*ACL 22*	68.57	55.99	60.64	19.42	-4.60	2.85
Compactor (Karimi Mahabadi, Henderson, and Ruder 2021)	*NIPS 21*	68.08	56.41	60.08	19.22	-4.55	2.78
Compactor++ (Karimi Mahabadi, Henderson, and Ruder 2021)	*NIPS 21*	67.26	55.69	59.47	19.54	-5.84	2.66
MULTI-TASK PEFT-BASED METHODS							
HyperFormer (Karimi Mahabadi et al. 2021)	*ACL 21*	71.43	60.73	65.54	17.77	+2.64	75.32
Polyhistor (Liu et al. 2022)	*NIPS 22*	70.87	59.54	65.47	17.47	+2.34	8.96
VMT-Adapter (Xin et al. 2024a)	*AAAI 24*	71.60	60.67	64.02	16.41	+3.96	3.68
MTLoRA (Agiza, Neseem, and Reda 2024)	*CVPR 24*	67.90	59.84	65.40	16.60	+2.55	8.34
DiTask (Mantri et al. 2025)	*CVPR 25*	70.09	59.03	64.55	17.47	+1.48	3.55
TADFormer (Baek et al. 2025) **(previous best)**	*CVPR 25*	70.82	60.45	**65.88**	16.48	+4.24	7.38
Ours	*AAAI 26*	**74.14**	61.75	65.69	**14.48**	**+8.70**	**2.36**

Table 1: Quantitative comparison on the PASCAL-Context dataset across four dense prediction tasks. We compare traditional fine-tuning methods, single-task PEFT-based methods, and multi-task PEFT-based methods. Our method achieves the best overall performance while using the fewest trainable parameters (#TP), demonstrating superior efficiency and effectiveness.

employs projection layers that reduce feature dimensions to $\frac{1}{r}$ of their original size, significantly compressing the parameter space. Importantly, only the bi-directional adapter components scale with the number of tasks, while the rest of the architecture remains task-independent, ensuring scalable multi-task adaptation. The MoTE module, in turn, is composed of lightweight fully connected layers. Under the default configuration with a Swin-Tiny backbone, our method introduces just 2.36M trainable parameters.

Second, we analyze the gradient computations involved in our method during the backpropagation process. Consider an N-layer neural network, where the output of each layer is denoted by $\mathbf{Y} = \{\mathcal{Y}_1, \mathcal{Y}_2, ..., \mathcal{Y}_N\}$, and the corresponding layer parameters are represented as $\mathbf{\Theta} = \{\theta_1, \theta_2, ..., \theta_N\}$. To minimize the loss function $\mathcal{L}$, the gradient with respect to the parameters θ_i is computed using the chain rule as follows:

$$\frac{\partial \mathcal{L}}{\partial \theta_i} = \underbrace{\frac{\partial \mathcal{L}}{\partial \mathcal{Y}_L}\frac{\partial \mathcal{Y}_L}{\partial \mathcal{Y}_{L-1}} \cdots \frac{\partial \mathcal{Y}_{i+1}}{\partial \mathcal{Y}_i}}_{\text{gradient w.r.t. intermediate layers}} \frac{\partial \mathcal{Y}_i}{\partial \theta_i} = \frac{\partial \mathcal{L}}{\partial \mathcal{Y}_i}\frac{\partial \mathcal{Y}_i}{\partial \theta_i}. \quad (12)$$

It can be observed that the partial derivative of $\mathcal{L}$ with respect to θ_i depends on the outputs of intermediate layers. Consequently, in backbone-integrated PEFT methods such as Adapters, LoRA, and prompt tuning, computing parameter gradients requires backpropagating through the backbone to obtain $\frac{\partial \mathcal{L}}{\partial \mathcal{Y}_i}$, which leads to considerable gradient computation and memory overhead, reducing training efficiency.

In contrast, our framework places adaptation modules in parallel to the backbone, effectively bypassing its intermediate layers during backpropagation. As a result, the computation of gradients avoids propagation through the backbone, leading to less GPU memory and reduced training time.

4 Experiment

4.1 Experimental Setups

Dataset and Metrics We evaluate our method on the PASCAL-Context dataset (Everingham et al. 2010), a large-scale benchmark for multi-task dense scene understanding. Following prior work (Liu et al. 2022; Baek et al. 2025; Agiza, Neseem, and Reda 2024), we consider four tasks: 21-class semantic segmentation, 7-class human part segmentation, surface normal estimation, and saliency estimation. The dataset includes 4,998 training and 5,105 testing images. To assess performance across tasks, we adopt a set of standard evaluation metrics. Mean Intersection over Union (mIoU) is used for semantic segmentation, human part segmentation, and saliency estimation. Root Mean Squared Error (RMSE) is employed for surface normal estimation. Additionally, we report $\Delta\mathbf{m}$ (Agiza, Neseem, and Reda 2024), which measures the relative overall improvement compared to the single-task full fine-tuning baseline:

$$\Delta\mathbf{m} = \frac{1}{K}\sum_{k=1}^{K}(-1)^{l_k}\frac{(M_k - M_{st,k})}{M_{st,k}}, \quad (13)$$

where M_k and $M_{st,k}$ denote the performance of our model and the single-task fine-tuning baseline on task t_k, respec-

Backbone	Method	#TP (M)↓	Memory (GB)↓	Time (s/batch)↓	$\Delta\mathbf{m}$ (%) ↑
Swin-Tiny	MTLoRA	8.34	18.22	**0.232**	+2.55
	DiTask	3.55	18.19	0.403	+1.48
	TADFormer	7.38	20.30	0.376	+4.24
	Ours	**2.36**	**9.92**	0.242	**+8.70**
Swin-Small	MTLoRA	13.11	20.58	0.321	+5.75
	DiTask	3.59	20.56	0.641	+4.68
	TADFormer	11.72	22.76	0.473	+6.27
	Ours	**2.36**	**10.03**	**0.266**	**+8.99**
Swin-Base	MTLoRA	17.76	23.91	0.411	+5.95
	DiTask	5.08	24.18	0.826	+6.52
	TADFormer	15.81	26.52	0.607	+7.53
	Ours	**4.13**	**15.32**	**0.322**	**+9.58**

Table 2: Comprehensive efficiency and performance comparison results with various backbone scales.

Method	SemSeg (mIoU)	HumPa (mIoU)	Saliency (mIoU)	Normals (RMSE)	$\Delta\mathbf{m}$ (%) ↑	#TP (M)↓
Pre-training Dataset = *Imagenet 1k*						
MTLoRA	67.90	59.84	65.40	16.60	+2.55	8.34
DiTask	70.09	59.03	64.55	17.47	+1.48	3.55
TADFormer	70.82	60.45	**65.88**	16.48	+4.24	7.38
Ours	**74.14**	**61.75**	65.69	**14.48**	**+8.70**	**2.36**
Pre-training Dataset = *Imagenet 22k*						
MTLoRA	68.71	61.32	65.37	16.44	+3.65	8.34
DiTask	69.66	62.02	65.00	17.00	+3.22	3.55
TADFormer	72.47	62.00	**65.99**	16.48	+5.56	7.38
Ours	**76.19**	**63.69**	65.56	**14.46**	**+10.22**	**2.36**

Table 3: Performance and trainable parameter comparison results under different pre-training dataset scales.

tively. $l_k = 1$ if lower is better for t_k, and 0 otherwise.

Implementation Details For fair comparison, all experiments are conducted using PyTorch on the same device, with Swin Transformer Tiny (*ImageNet-1k* pre-trained) as the default backbone. Following the loss setup in (Lu et al. 2024), models are trained for 300 epochs using AdamW optimizer, with a per-GPU batch size of 8, a learning rate of 4×10^{-4}, weight decay of 1×10^{-4}, and a cosine learning rate schedule. The reduction factor r is set to 4 by default.

4.2 Baselines

We compare our method against a wide range of baselines, categorized into three groups: **(1)** Conventional fine-tuning methods, which update the full or partial pre-trained weights, including single-task full fine-tuning, multi-task full fine-tuning, and decoder-only fine-tuning. **(2)** Single-task PEFT methods, including Adapter (He et al. 2022), VL-Adapter (Yi-Lin Sung 2022), LoRA (Hu et al. 2022a), VPT (Jia et al. 2022), BitFit (Zaken, Goldberg, and Ravfogel 2022), Compactor and Compactor++(Karimi Mahabadi, Henderson, and Ruder 2021). **(3)** Multi-task PEFT methods, including HyperFormer (Karimi Mahabadi et al. 2021), Polyhistor (Liu et al. 2022), VMT-Adapter (Xin et al. 2024a), MTLoRA (Agiza, Neseem, and Reda 2024), TADFormer (Baek et al. 2025), and DiTask (Mantri et al. 2025).

BDTA	MoTE	SemSeg (mIoU)	HumPa (mIoU)	Saliency (mIoU)	Normals (RMSE)	$\Delta\mathbf{m}$ (%) ↑	#TP (M)↓
✗	✗	67.21	61.93	62.35	17.97	0	112.62
✔	✗	73.92	61.30	**65.78**	14.52	+8.40	1.96
✔	✔	**74.14**	**61.75**	65.69	**14.48**	**+8.70**	2.36

Table 4: Ablation study on the effectiveness of BDTA and MoTE modules in proposed method.

Adaptation Configure	SemSeg (mIoU)	HumPa (mIoU)	Saliency (mIoU)	Normals (RMSE)	$\Delta\mathbf{m}$ (%) ↑	#TP (M)↓
G2T-only	73.63	60.92	63.96	15.11	+6.59	2.11
T2G-only	71.97	59.58	64.55	14.42	+6.64	2.11
T2G+G2T	**74.10**	**60.92**	**65.05**	**14.66**	**+8.70**	2.36

Table 5: Ablation study of different task adaptation configurations in BDTA module.

4.3 Main Results

Tab. 1 reports task-wise results, overall metric $\Delta\mathbf{m}$, and trainable parameters (#**TP**) for all methods. Traditional fine-tuning strategies either demand large parameter counts (e.g., single-/multi-task full fine-tuning) or exhibit lower robustness when tuning only decoders. While single-task PEFT approaches are parameter-efficient, none yields a positive $\Delta\mathbf{m}$, exposing limitations in multi-task settings. In contrast, our method achieves both the lowest #**TP** (2.36M) and the best overall performance. It ranks first or second on three of four tasks, with significant improvements of +2.54 mIoU in semantics segmentation and +1.93 RMSE in normal estimation. Overall, our method achieves a $\Delta\mathbf{m}$ score of +8.70%, outperforming the second-best method, TADFormer (4.24%), by +4.46%. This corresponds to a 2.05× performance improvement, with only 32% of its trainable parameters. We next provide a more comprehensive efficiency analysis beyond trainable parameters.

Tab. 2 compares the overall training efficiency of our method with recent state-of-the-art multi-task PEFT approaches (MTLoRA, DiTask, and TADFormer) under identical settings—same input size, numerical precision, and hardware—across various backbone scales. Metrics include trainable parameters (#**TP**), GPU memory, training time, and task performance. Existing PEFT methods insert adaptation modules serially within the backbone, leading to higher memory usage due to gradient flow through intermediate backbone layers. They also require dense tuning of all attention and FFN layers, whereas our method only adapts intermediate output features of each encoder stage, significantly reducing #**TP**—especially with larger backbones. In terms of training time, our method is the fastest overall, with only a slight delay compared to MTLoRA on Swin-Tiny. With the default Swin-Tiny backbone, our method achieves over 2× improvement in $\Delta\mathbf{m}$ compared to the best-performing

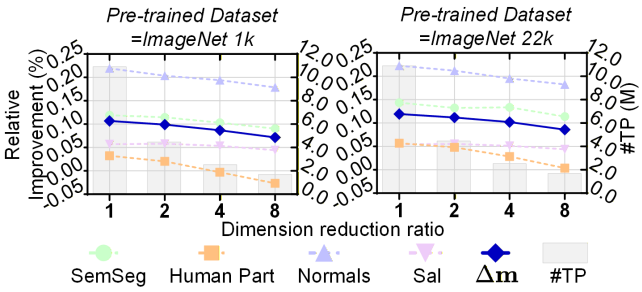

Figure 3: Impact of the reduction factor r on parameter efficiency and performance with different pre-training datasets.

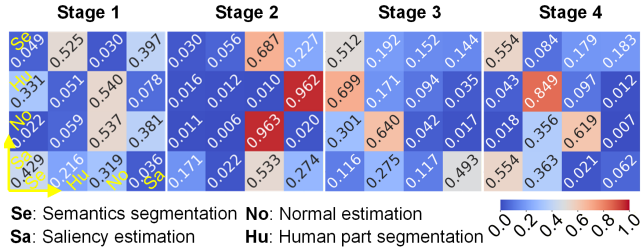

Figure 4: The average gating scores learned by different task routers in MoTE modules of different encoding stages.

TADFormer, while using only $\sim$ 30% of the parameters, $\sim$ 50% of the GPU memory, and $\sim$ 60% of the training time. Overall, our method demonstrates the best efficiency and effectiveness, notably outperforming all baselines.

Moreover, model performance also scales with the size of the pretraining dataset. As shown in Tab. 3, we compare MTLoRA, DiTask, TADFormer, and our method under varying pretraining data sizes. Results show that larger pretraining datasets lead to better performance. Notably, our method consistently surpasses all baselines across both scales.

4.4 In-depth analysis

Ablation study We conduct ablations to evaluate the contributions of BDTA and MoTE. As shown in Tab. 4, we compare three variants: (1) removing both components (single-task full fine-tuning baseline), (2) using BDTA only, and (3) using both BDTA and MoTE. BDTA alone significantly improves performance and reduces trainable parameters by effectively adapting pretrained features. Adding MoTE further enhances performance by enabling cross-task interaction.

BDTA Analysis The BDTA module extracts task-adapted features via a bidirectional adaptation paradigm that decouples representations into task-generic and task-specific components. To validate this design, we ablate two one-way variants: *T2G* and *G2T*. As shown in Tab. 5, the full bidirectional setup performs best, followed by *G2T*, while *T2G* performs worst. This is because *G2T* captures task-specific nuances but lacks shared knowledge, whereas *T2G* shares features across tasks but lacks specificity. Their combination enables the model to learn both shared and distinct representations, yielding better results.

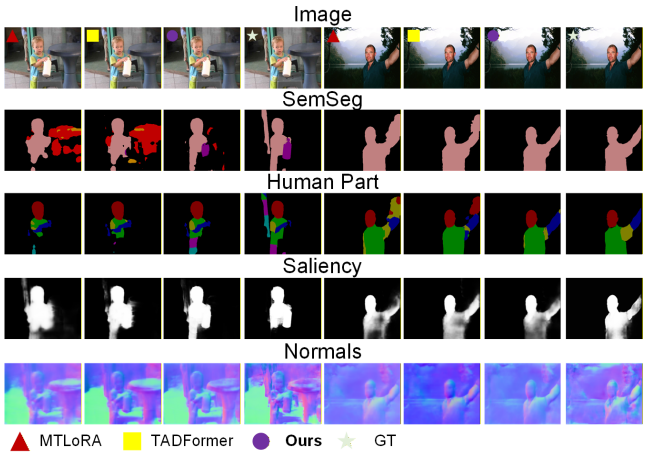

Figure 5: Qualitative multi-task prediction results comparing MTLoRA, TADFormer, and our method.

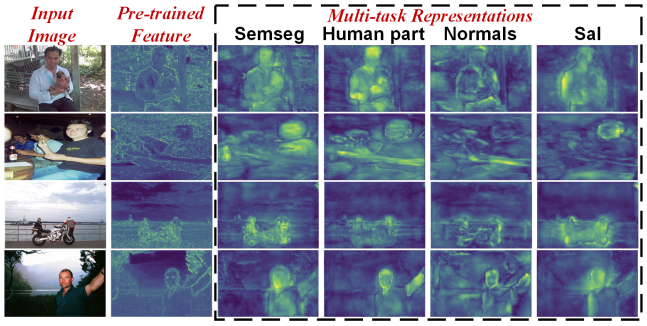

Figure 6: Visualization of pretrained within GMB and task-specific adapted feature maps for different tasks within TBB.

To ensure lightweight adaptation, BDTA uses low-rank projections reducing feature dimensions by a factor of r. We analyze the impact of varying r in Fig. 3. Smaller r improves accuracy but increases computational cost. We set $r = 4$ by default to balance performance and efficiency. It is worth noting that the adjustable nature of r allows for flexible configuration according to deployment requirements.

MoTE analysis After obtaining task-adapted features via BDTA, MoTE is employed to facilitate cross-task interactions. Fig. 4 visualizes the average gating scores learned by MoTE across different stages and tasks on the PASCAL-Context dataset, where each row corresponds to the gating results of one task router. The observation that each task assigns high gating scores to others indicates a mutual enhancement mechanism, highlighting the effectiveness of cross-task interaction.

Visualization To provide an intuitive comparison, Fig. 5 presents visualizations of multi-task prediction results to illustrate the performance improvements of our proposed method. We benchmark our approach against recent state-of-the-art methods, MTLoRA and TADFormer. The results show that our method consistently produces more accurate

and semantically reasonable predictions across all tasks.

To better illustrate the adaptation effectiveness of our method on downstream tasks, Fig. 6 visualizes the pretrained features from the GMB and the task-specific features from the TBB for different tasks. The pretrained features mainly capture low-level semantics, such as edges, while the adapted features exhibit stronger task-specific semantic representations. For example, the features for human part segmentation focus more on body limbs, whereas saliency features highlight visually prominent regions in the image. These results clearly demonstrate the effectiveness of the proposed feature adaptation mechanism.

5 Conclusions

This paper presents a comprehensively efficient MTL framework for dense vision adaptation. We first identify a key limitation of existing PEFT-based methods—their narrow focus on parameter efficiency—and extend the notion of efficiency to also include GPU memory usage and training time. To this end, we design a dual-branch architecture, where the pretrained backbone serves as a task-generic main branch, and the proposed BDTA modules are connected in parallel to form the task bypass branch for feature adaptation. Furthermore, we introduce the MoTE module to enable effective cross-task interaction. We validate our approach on four representative dense visual prediction tasks. The results demonstrate that our method not only significantly improves prediction performance but also substantially reduces parameters, GPU memory consumption, and training time. Future work will explore efficient model adaptation for more complex MTL scenarios with 2D and 3D multimodal visual data.

References

Agiza, A.; Neseem, M.; and Reda, S. 2024. MTLoRA: Low-Rank Adaptation Approach for Efficient Multi-Task Learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 16196–16205.

Awais, M.; Naseer, M.; Khan, S.; Anwer, R. M.; Cholakkal, H.; Shah, M.; Yang, M.-H.; and Khan, F. S. 2025. Foundation Models Defining a New Era in Vision: a Survey and Outlook. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Baek, S.; Lee, S.; Jo, H.; Choi, H.; and Min, D. 2025. TADFormer: Task-Adaptive Dynamic Transformer for Efficient Multi-Task Learning. *arXiv preprint arXiv:2501.04293*.

Caron, M.; Touvron, H.; Misra, I.; Jégou, H.; Mairal, J.; Bojanowski, P.; and Joulin, A. 2021. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, 9650–9660.

Chen, S.; Ge, C.; Tong, Z.; Wang, J.; Song, Y.; Wang, J.; and Luo, P. 2022. Adaptformer: Adapting vision transformers for scalable visual recognition. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Chen, Z.; Duan, Y.; Wang, W.; He, J.; Lu, T.; Dai, J.; and Qiao, Y. 2023a. Vision transformer adapter for dense predictions. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Chen, Z.; Duan, Y.; Wang, W.; He, J.; Lu, T.; Dai, J.; and Qiao, Y. 2023b. Vision Transformer Adapter for Dense Predictions. In *International Conference on Learning Representations (ICLR)*.

Ding, N.; Qin, Y.; Yang, G.; Wei, F.; Yang, Z.; Su, Y.; Hu, S.; Chen, Y.; Chan, C.-M.; Chen, W.; et al. 2023. Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nature Machine Intelligence*, 5(3): 220–235.

Everingham, M.; Van Gool, L.; Williams, C. K.; Winn, J.; and Zisserman, A. 2010. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision (IJCV)*.

Han, C.; Wang, Q.; Cui, Y.; Cao, Z.; Wang, W.; Qi, S.; and Liu, D. 2023. Eˆ 2vpt: An effective and efficient approach for visual prompt tuning. *arXiv preprint arXiv:2307.13770*.

Han, Z.; Gao, C.; Liu, J.; Zhang, J.; and Zhang, S. Q. 2024. Parameter-efficient fine-tuning for large models: A comprehensive survey. *arXiv preprint arXiv:2403.14608*.

He, J.; Zhou, C.; Ma, X.; Berg-Kirkpatrick, T.; and Neubig, G. 2022. Towards a Unified View of Parameter-Efficient Transfer Learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Houlsby, N.; Giurgiu, A.; Jastrzebski, S.; Morrone, B.; De Laroussilhe, Q.; Gesmundo, A.; Attariyan, M.; and Gelly, S. 2019. Parameter-efficient transfer learning for NLP. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2790–2799. PMLR.

Hu, E.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, L.; and Chen, W. 2022a. LoRA: Low-Rank Adaptation of Large Language Models. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Hu, E. J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; Chen, W.; et al. 2022b. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2): 3.

Jia, M.; Tang, L.; Chen, B.-C.; Cardie, C.; Belongie, S.; Hariharan, B.; and Lim, S.-N. 2022. Visual Prompt Tuning. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

Jie, S.; Deng, Z.-H.; Chen, S.; and Jin, Z. 2024. Convolutional bypasses are better vision transformer adapters. In *ECAI 2024*, 202–209. IOS Press.

Kaplan, J.; McCandlish, S.; Henighan, T.; Brown, T. B.; Chess, B.; Child, R.; Gray, S.; Radford, A.; Wu, J.; and Amodei, D. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Karimi Mahabadi, R.; Henderson, J.; and Ruder, S. 2021. Compacter: Efficient low-rank hypercomplex adapter layers. *Advances in Neural Information Processing Systems (NeurIPS)*, 34: 1022–1035.

Karimi Mahabadi, R.; Ruder, S.; Dehghani, M.; and Henderson, J. 2021. Parameter-efficient Multi-task Fine-tuning for Transformers via Shared Hypernetworks. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Kirillov, A.; Mintun, E.; Ravi, N.; Mao, H.; Rolland, C.; Gustafson, L.; Xiao, T.; Whitehead, S.; Berg, A. C.; Lo, W.-Y.; et al. 2023. Segment anything. In *Proceedings of the IEEE/CVF international conference on computer vision*, 4015–4026.

Li, Y.; Fan, H.; Hu, R.; Feichtenhofer, C.; and He, K. 2023. Scaling language-image pre-training via masking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 23390–23400.

Liu, Y.-C.; Ma, C.-Y.; Tian, J.; He, Z.; and Kira, Z. 2022. Polyhistor: Parameter-Efficient Multi-Task Adaptation for Dense Vision Tasks. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Liu, Z.; Lin, Y.; Cao, Y.; Hu, H.; Wei, Y.; Zhang, Z.; Lin, S.; and Guo, B. 2021. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

Lu, Y.; Sirejiding, S.; Bayramli, B.; Huang, S.; Ding, Y.; and Lu, H. 2024. Task Indicating Transformer for Task-Conditional Dense Predictions. In *ICASSP*, 3625–3629. IEEE.

Mantri, K. S. I.; Schönlieb, C.-B.; Ribeiro, B.; Baskin, C.; and Eliasof, M. 2025. DiTASK: Multi-Task Fine-Tuning with Diffeomorphic Transformations. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, 25218–25229.

Mercea, O.-B.; Gritsenko, A.; Schmid, C.; and Arnab, A. 2024. Time-memory-and parameter-efficient visual adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 5536–5545.

Niu, S.; Liu, Y.; Wang, J.; and Song, H. 2021. A decade survey of transfer learning (2010–2020). *IEEE Transactions on Artificial Intelligence*, 1(2): 151–166.

Oquab, M.; Darcet, T.; Moutakanni, T.; Vo, H.; Szafraniec, M.; Khalidov, V.; Fernandez, P.; Haziza, D.; Massa, F.; El-Nouby, A.; et al. 2023. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*.

Pfeiffer, J.; Rücklé, A.; Poth, C.; Kamath, A.; Vulić, I.; Ruder, S.; Cho, K.; and Gurevych, I. 2020. AdapterHub: A Framework for Adapting Transformers. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Shazeer, N.; Mirhoseini, A.; Maziarz, K.; Davis, A.; Le, Q.; Hinton, G.; and Dean, J. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*.

Sung, Y.-L.; Cho, J.; and Bansal, M. 2022. Lst: Ladder side-tuning for parameter and memory efficient transfer learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 35: 12991–13005.

Xia, C.; Wang, X.; Lv, F.; Hao, X.; and Shi, Y. 2024. Vitcomer: Vision transformer with convolutional multi-scale feature interaction for dense predictions. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 5493–5502.

Xie, E.; Wang, W.; Yu, Z.; Anandkumar, A.; Alvarez, J. M.; and Luo, P. 2021. SegFormer: Simple and efficient design for semantic segmentation with transformers. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Xin, Y.; Du, J.; Wang, Q.; Lin, Z.; and Yan, K. 2024a. Vmt-adapter: Parameter-efficient transfer learning for multi-task dense scene understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 38, 16085–16093.

Xin, Y.; Yang, J.; Luo, S.; Zhou, H.; Du, J.; Liu, X.; Fan, Y.; Li, Q.; and Du, Y. 2024b. Parameter-efficient fine-tuning for pre-trained vision models: A survey. *arXiv preprint arXiv:2402.02242*.

Xu, Y.; Li, X.; Yuan, H.; Yang, Y.; and Zhang, L. 2023. Multi-task learning with multi-query transformer for dense prediction. *IEEE Transactions on Circuits and Systems for Video Technology*.

Yang, L.; Kang, B.; Huang, Z.; Xu, X.; Feng, J.; and Zhao, H. 2024. Depth anything: Unleashing the power of large-scale unlabeled data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 10371–10381.

Yi-Lin Sung, M. B., Jaemin Cho. 2022. VL-Adapter: Parameter-Efficient Transfer Learning for Vision-and-Language Tasks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Yin, D.; Han, X.; Li, B.; Feng, H.; and Bai, J. 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*, 1398–1406.

Zaken, E. B.; Goldberg, Y.; and Ravfogel, S. 2022. BitFit: Simple Parameter-efficient Fine-tuning for Transformer-based Masked Language-models. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Zhou, K.; Yang, J.; Loy, C. C.; and Liu, Z. 2022. Learning to prompt for vision-language models. *International Journal of Computer Vision*, 130(9): 2337–2348.