Dual DTL: Double-Side Compact Transfer Networks for Efficient ViT Adaptation

Zonghao Ding[1,2], Minghao Fu[1,2], and Jianxin Wu[1,2(✉)]

[1]National Key Laboratory for Novel Software Technology, Nanjing University, China
[2]School of Artificial Intelligence, Nanjing University, China
dingzh@lamda.nju.edu.cn, fumh@lamda.nju.edu.cn, wujx2001@gmail.com

Abstract. We propose **DDTL** (Double Disentangled Transfer Learning), a novel parameter-efficient framework for adapting vision transformers. DDTL enhances the recently introduced DTL by incorporating dual compact side networks, placed before and in each transformer block, enabling more expressive and fine-grained task adaptation. Unlike traditional PETL approaches that entangle the backbone layers, DDTL remains highly memory-efficient while achieving superior accuracy. We further explore block-dropping strategies, including loss-guided and cosine similarity-based methods, to reduce computation cost during training. Experiments on VTAB-1K across multiple ViT scales demonstrate that DDTL consistently outperforms strong PETL baselines with low parameter increase and memory footprint. Our method provides a scalable and flexible solution for efficient transfer learning in low-resource settings.

Keywords: Transfer Learning · Efficient Training · Image Recognition · Block Pruning.

1 Introduction

In recent years, pre-trained vision models have demonstrated remarkable generalization and transfer capabilities across a wide range of computer vision tasks [3,6,20,16]. These models are typically pre-trained on large-scale datasets and subsequently adapted to downstream tasks through fine-tuning. However, as model sizes continue to grow, full fine-tuning faces two major challenges: first, the computational cost and time required to update all parameters have increased significantly; second, the need to cache intermediate activations and gradients during backpropagation imposes heavy GPU memory overhead, which limits scalability and practicality, especially in memory-constrained settings.

To address these issues, Parameter-Efficient Transfer Learning (PETL) methods [1,8,9,14,22] have been proposed. These approaches introduce a small number of trainable parameters or reparameterization mechanisms to effectively adapt large pre-trained models to target tasks while keeping most of the backbone frozen. PETL methods not only drastically reduce computational and storage

costs, but also achieve performance comparable to, or even better than, full fine-tuning in many scenarios. As a result, PETL has become a mainstream paradigm for adapting large-scale pre-trained models.

Despite significantly reducing the number of trainable parameters, existing PETL methods often exhibit only moderate reductions in GPU memory usage—typically around 25%. This limitation remains a bottleneck for adapting large models in memory-constrained environments. Most PETL approaches embed small trainable modules into a frozen backbone, resulting in tightly coupled architectures. As shown in [18], training such coupled systems requires caching intermediate activations and gradients to ensure correct parameter updates, leading to substantial memory consumption and limiting the practical deployment of large models. To address this, Fu et al. [5] proposed Disentangled Transfer Learning (DTL), which introduces a lightweight Compact Side Network (CSN) to decouple weight updates from the main backbone. This design significantly reduces memory usage while maintaining high adaptation performance.

However, the original DTL architecture inserts a single CSN only before each Transformer block, overlooking the intermediate features within the block, namely those from the Multi-Head Self-Attention (MHSA) module and the Feed-Forward Network (FFN). While the attention module captures rich semantic correlations, the FFN performs nonlinear transformations on features; insufficient decoupling of these two stages can limit the effectiveness of task-specific adaptation.

To overcome this limitation, we propose **Dual DTL (DDTL)**, as shown in Figure 1, a novel framework that enables fine-grained disentanglement through a dual-branch CSN design. Specifically, DDTL inserts low-rank linear mappings before each ViT block and after the attention module. This dual-side CSN allows separate extraction of semantic features from attention and nonlinear features from FFN, thereby achieving intra-block dual disentanglement and enhancing the expressiveness of adaptation.

In addition, we explore model slimming through block pruning and introduce two block dropping strategies. The first strategy employs a loss-guided evaluation, where each block is temporarily dropped and the loss is computed on a batch of images to identify the least critical block. The second strategy relies on cosine similarity to evaluate feature redundancy between adjacent blocks, removing the latter in highly similar pairs. These strategies effectively reduce computational cost while preserving accuracy, providing a more scalable and efficient transfer learning paradigm.

2 Related Work

Parameter-Efficient Transfer Learning (PETL) has emerged as a prominent approach for adapting large-scale pre-trained models to downstream tasks. As the size of foundation models continues to grow, the computational, memory, and storage costs associated with full fine-tuning have become increasingly prohibitive. PETL offers an efficient alternative by freezing most of the pre-trained

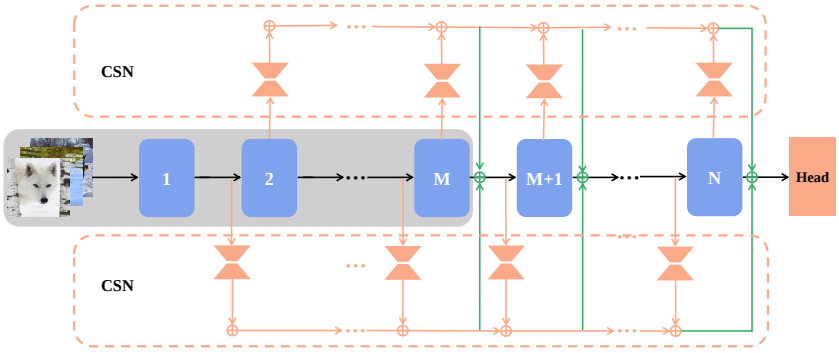

Fig. 1. Architecture of the proposed Dual Disentangled Transfer Learning (DDTL). The ViT backbone consists of multiple Transformer blocks. For each block (starting from the second), DDTL inserts two Compact Side Networks (CSNs): a Pre-CSN before the block input and a Post-CSN after the self-attention module. Each CSN uses low-rank linear layers to extract task-specific features(orange arrows). Optional depthwise separable convolutions further refine the adapted signals before merging them back into the backbone via residual connections (green arrows). During fine-tuning, only parameters of the CSN module and the task-specific classification head are updated (illustrated in orange). Best viewed in color.

parameters while only adapting a small, task-specific set of trainable components.

Initially proposed for natural language processing (NLP) (NLP) [7,12,13,8], PETL methods have since been successfully extended to computer vision with the advent of Vision Transformers (ViTs) [3,16]. The core principle of PETL is to retain the backbone model frozen and introduce lightweight, trainable modules or learnable prompts to facilitate task adaptation.

Adapter-based methods. Adapter methods insert small bottleneck modules within Transformer blocks, typically consisting of a down-projection and up-projection linear layer [7,1]. For example, AdaptFormer [1] adds a parallel branch with two linear layers and a ReLU activation to the MLP module, optimizing only this branch while freezing the rest of the network. RepAdapter [17] sequentially inserts lightweight modules into the MHA and MLP blocks, which can later be re-parameterized into nearby projection weights for inference efficiency. Dyn-Adapter [23] introduces a dynamic path selection mechanism based on multi-layer feature extraction and bi-sparse regularization, reducing inference FLOPs by up to 50% without accuracy degradation.

Prompt-based methods. In the field of NLP, Prompt-Tuning[12], Prefix-Tuning[13] and P-Tuning[15] perform well. Vision Prompt Tuning (VPT) [9]

inserts learnable tokens into the ViT input sequence, which interact with image patches through self-attention to encode task-specific representations.

LoRA-based methods. LoRA [8] approximates full-rank weight updates using low-rank decomposition. It learns updates via a pair of small matrices and merges them back into the original weights during inference. QLoRA[2] enhances this with quantization. ALoRE [4] expands this idea by introducing multi-branch designs and Kronecker-product-based hypercomplex spaces to aggregate low-rank experts.

Side-network-based methods. Side-Tuning[21] performs transfer learning by training a lightweight side network and fusing its output with a frozen backbone model at the final layer. LST[18] further utilizes the multi-layer intermediate features of the backbone network as inputs to the side network. DTL [5] inserts compact side networks (CSNs) before Transformer blocks to extract task-specific features in a memory-efficient manner, and appropriately add the information back to the backbone network .

Other methods. Bitfit[19] adjusts only bias terms in the backbone. SSF [14] applies a linear transformation to intermediate features. FacT [10] utilizes tensor decomposition to reduce the parameter count in PETL modules. NOAH [22] combines several PETL strategies into a supernet. S2A[11] combines structural modules with 4-bit activation quantization.

3 Methodology

3.1 Review of DTL

Disentangled Transfer Learning (DTL) alleviates the high GPU memory cost of traditional fine-tuning by decoupling task-specific parameter updates from the frozen backbone. It introduces a **Compact Side Network (CSN)**, inserted before selected Transformer blocks. The CSN progressively aggregates task-specific knowledge using lightweight low-rank linear layers, and re-injects this information back into later layers to guide adaptation.

For a Vision Transformer (ViT) with N blocks, let $x \in \mathbb{R}^{(n+1)\times d}$ be the input tokens and z_i be the output of the i-th block b_i. Denote $W_i = a_i c_i \in \mathbb{R}^{d\times d}$ as the weight matrix accounting for the i-th block, with $a_i \in \mathbb{R}^{d\times d'}$, $c_i \in \mathbb{R}^{d'\times d}$ and $d' \ll d$ .In DTL, the CSN performs:

$$h_{i+1} = h_i + z_i W_i \,, \tag{1}$$

$$z_{i+1} = b_i \,, \tag{2}$$

$$z'_{i+1} = \begin{cases} z_{i+1} + \theta(h_{i+1}) & \text{if } \ i \geq M \\ z_{i+1} & \text{otherwise} \,, \end{cases} \tag{3}$$

where $\theta(\cdot)$ denotes the Swish activation function.

3.2 DDTL: Double Disentangled Transfer Learning

We illustrate the overall architecture of the proposed **DDTL** in Figure 1. The DDTL framework builds on a Vision Transformer (ViT) backbone and introduces a dual-branch compact side network design to enable fine-grained feature adaptation. Specifically, for each Transformer block (starting from the second), we insert two lightweight CSNs:

- The **Pre-CSN** is placed before the Transformer block and operates on its input token sequence.
- The **Post-CSN** is applied to the output of the self-attention module within the same block.

The updated design is defined as:

$$h_{i+1}^{\text{pre}} = h_i^{\text{pre}} + z_i W_i^{\text{pre}}\,, \tag{4}$$

$$h_{i+1}^{\text{post}} = h_i^{\text{post}} + \text{SA}_i(z_i) W_i^{\text{post}}\,, \tag{5}$$

$$z_{i+1} = b_i(z_i)\,, \tag{6}$$

$$z'_{i+1} = \begin{cases} z_{i+1} + \theta(h_i^{\text{post}}) + \theta(h_i^{\text{pre}}) & \text{if } \ i \geq M \\ z_{i+1} & \text{otherwise}\,. \end{cases} \tag{7}$$

Optionally, the Swish outputs may pass through a global depthwise separable convolution (DWConv) layer $g(\cdot)$:

$$z'_{i+1} = \begin{cases} z_{i+1} + g(\theta(h_i^{\text{post}})) + g(\theta(h_i^{\text{pre}})) & \text{if } \ i \geq M \\ z_{i+1} & \text{otherwise}\,. \end{cases} \tag{8}$$

New Design Insight: Starting from Block 2 Empirical results show that DDTL consistently performs better when dual CSNs start from the second block ($i = 2$), rather than the first.

Why skip the first block? The first block in ViT typically handles generic embedding alignment. Applying adaptation at this early stage may inject noise into globally shared token features.

3.3 Theoretical Analysis

Memory Efficiency In DDTL, backpropagation through the backbone is only enabled from block M onward. For blocks before M, the backbone remains completely frozen and disentangled from the learning process. Gradients in these early stages are propagated solely through the Compact Side Networks (CSNs), without traversing the backbone layers, effectively reducing memory usage and computational overhead.

$$\text{Cached intermediates} \propto \sum_{i=M}^{N} \|\nabla_{b_i}\| + \sum_{i=2}^{N} \left(\|\nabla_{W_i^{\text{pre}}}\| + \|\nabla_{W_i^{\text{post}}}\| \right) , \quad (9)$$

The parameter count of side network W_i^{pre} and W_i^{post} are much smaller than that of backbone network b_i. This leads to up to about 50% savings in GPU memory.

Parameter Complexity Each CSN introduces $d \times d'$ parameters. With two CSNs per block and $(N-1)$ adapted blocks (from block 2 to N), the total added parameters are:

$$\text{CSN params} = 2 \times 2 \times (N-1) \times d \times d' . \quad (10)$$

Additionally, the global depthwise separable convolution introduces a small number of parameters. For a convolution of kernel size 5 and dimension d, it contributes:

$$\text{Depthwise params} = d \times (5 \times 5) , \quad (11)$$

For ViT-B/16 with $N = 12$, $d = 768$, $d' = 2$, this becomes only approximately 0.09M.

3.4 Block Dropping Strategies

To reduce computation and training time, we propose two block dropping strategies:

Strategy 1: Loss-Guided Block Dropping In every drop interval, remove each candidate block one at a time. Evaluate the model on one batch data and compute the loss. Drop the block whose removal leads to the smallest increase (or largest decrease) in loss.

Strategy 2: Cosine Similarity-Based Dropping For each adjacent block pair, compute cosine similarity between their outputs. The block following the pair with the highest similarity is dropped.

Drop Schedule

- **Warm-up Phase**: No block is dropped during the first 10 epochs.
- **Drop Phase**: Starting from epoch 11, drop one block every 5 epochs.
- **Constraint**: Adjust M dynamically to ensure approximately half of backbone remain entangled with the CSNs.

These strategies help DDTL achieve scalable efficiency without sacrificing representational power.

	#param (M)	GPU mem (GB)	Natural							Specialized				Structured								Average
			Cifar100	Caltech101	DTD	Flower102	Pets	SVHN	Sun397	Camelyon	EuroSAT	Resisc45	Retinopathy	Clevr-Count	Clevr-Dist	DMLab	KITTI-Dist	dSpr-Loc	dSpr-Ori	sNORB-Azim	sNORB-Ele	
Traditional Fine-Tuning																						
Full	85.8	4.7	68.9	87.7	64.3	97.2	86.9	87.4	38.8	79.7	95.7	84.2	73.9	56.3	58.6	41.7	65.5	57.5	46.7	25.7	29.1	68.9
Linear	0	0.6	64.4	85.0	63.2	97.0	86.3	36.6	51.0	78.5	87.5	68.5	74.0	34.3	30.6	33.2	55.4	12.5	20.0	9.6	19.2	57.6
PETL methods																						
BitFit	0.10	2.9	72.8	87.0	59.2	97.5	85.3	59.9	51.4	78.7	91.6	72.9	69.8	61.5	55.6	32.4	55.9	66.6	40.0	15.7	25.1	65.2
VPT	0.56	4.2	**78.8**	90.8	65.8	98.0	88.3	78.1	49.6	81.8	96.1	83.4	68.4	68.5	60.0	46.5	72.8	73.6	47.9	32.9	37.8	72.0
LST	2.38	2.7	59.5	91.5	69.0	99.2	89.9	79.5	54.6	86.9	95.9	85.3	74.1	81.8	61.8	52.2	81.0	71.7	49.5	33.7	45.2	74.3
LoRA	0.29	3.0	67.1	91.4	69.4	98.8	90.4	85.3	54.0	84.9	95.3	84.4	73.6	**82.9**	**69.2**	49.8	78.5	75.7	47.1	31.0	44.0	74.5
AdaptFormer	0.16	2.8	70.8	91.2	70.5	99.1	90.9	86.6	54.8	83.0	95.8	84.4	**76.3**	81.9	64.3	49.3	80.3	76.3	45.7	31.7	41.1	74.7
NOAH	0.43	3.3	69.6	92.7	70.2	99.1	90.4	86.1	53.7	84.4	95.4	83.9	75.8	82.8	68.9	49.9	81.7	81.8	48.3	32.8	44.2	75.5
FacT	0.07	3.9	70.6	90.6	70.8	99.1	90.7	**88.6**	54.1	84.8	96.2	84.5	75.7	82.6	68.2	49.8	80.7	80.8	47.4	33.2	43.0	75.6
SSF	0.21	4.9	69.0	92.6	**75.1**	99.4	**91.8**	90.2	52.9	87.4	95.9	87.4	75.5	75.9	62.3	**53.3**	80.6	77.3	54.9	29.5	37.9	75.7
DTL*	0.05	1.7	70.4	95.1	71.5	99.4	91.8	87.5	56.8	87.7	96.6	86.9	74.7	81.6	65.1	51.3	82.3	97.2	54.9	36.0	49.3	77.7
DTL	**0.05**	**1.8**	69.7	**94.9**	71.6	99.3	91.6	87.2	**56.7**	87.2	**96.3**	86.6	74.7	81.6	65.1	51.4	80.3	**97.0**	55.0	36.5	**49.2**	77.4
DDTL	0.09	2.0	69.7	94.1	70.7	**99.4**	91.0	87.6	56.1	**87.5**	96.0	**87.5**	74.7	82.7	64.0	51.1	**82.0**	96.6	**55.7**	**38.5**	48.0	**77.5**

Table 1. Results on the VTAB-1K benchmark with ViT-B/16 as the backbone. "#param" denotes the number of trainable parameters. "GPU mem" specifies the peak GPU memory footprint when fine-tuning with batch size 32. "Average" is the group-wise average accuracy over three groups. DTL* refers to the results reported in the original paper [5], while DTL denotes our reproduced implementation. The best results among PETL methods are in bold face.

4 Experiments

We conduct a comprehensive set of experiments to evaluate the effectiveness of our proposed method. First, we compare the performance of DDTL against other PETL baselines on the VTAB-1K [20] benchmark using a ViT-B [3] backbone. We then further assess DDTL across ViTs of different scales to analyze its scalability. Next, we investigate the impact of our two block-dropping strategies—cosine similarity-based and loss-guided dropping. Finally, we perform ablation studies to understand the contribution of each component.

4.1 Experimental Setup

Datasets. We evaluate our method on the VTAB-1K benchmark, which is designed to measure the generalization ability of representation learning methods. VTAB-1K consists of 19 diverse datasets grouped into three categories:

1. natural images captured by standard cameras,
2. specialized images from professional equipment,
3. structured images generated in synthetic environments.

These datasets span a variety of vision tasks, such as object recognition, counting, and depth estimation. Each dataset provides only 1,000 images for training, making VTAB-1K a challenging benchmark for evaluating PETL methods.

Baselines. We compare DDTL against a wide range of baselines. First, we include two common fine-tuning paradigms:

- **Full-tuning:** updates all parameters of the pre-trained backbone.
- **Linear probing (Fixed):** only trains a classification head while keeping the backbone frozen.

In addition, we benchmark against several strong PETL methods: **BitFit**[19], **VPT** [9], **LST**[18], **AdaptFormer**[1], **LoRA**[8], **NOAH** [22], **FacT** [10], and **SSF** [14].

Implementation Details. We adopt the AdamW optimizer and a cosine learning rate scheduler. Unless otherwise specified, all ViT backbones are pre-trained on ImageNet-21K. For each CSN, the rank of the low-rank linear mapping is set to $d' = 2$. The CSNs information is added back to the backbone starting from block $M = 7$ for all DTL-based methods using the ViT-B backbone. For larger or smaller backbones, we apply a proportional rule.

4.2 Main Results

Experimental Results on VTAB-1K. Table 1 presents a comprehensive comparison of parameter-efficient transfer learning (PETL) methods on the

VTAB-1K benchmark using ViT-B/16 as the backbone. Our proposed methods, DDTL, achieve state-of-the-art performance across multiple dimensions.

The trainable parameter of DTL is 0.05M, while that of DDTL is 0.09M—significantly fewer than existing PETL approaches such as VPT (0.56M), LoRA (0.29M), and AdaptFormer (0.16M)—both DTL and DDTL achieve top-tier average accuracy of 77.4% and 77.5%, outperforming all baselines by a notable margin. Moreover, they exhibit superior GPU memory efficiency, with DTL requiring only 1.8GB, which is 62% less than SSF and 43% less than LoRA. Although DDTL introduces additional side networks, it increases GPU usage by just 0.2 GB — remaining substantially more efficient than competing PETL approaches.

In terms of group-wise performance, DDTL demonstrates robust generalization. On Natural tasks, DTL and DDTL perform close to SSF and outperform other methods. In the Specialized category, DDTL obtains 86.4%, outperforming NOAH (84.9%) and AdaptFormer (84.4%). Crucially, DDTL delivers the best results in the Structured category, reaching 66.5% accuracy—highlighting its effectiveness in tasks requiring fine-grained representation learning such as Clevr, KITTI, and dSprites. Specifically, among the 19 VTAB-1K datasets, DTL achieves the best top-1 accuracy on 5 datasets, while DDTL leads on 6 datasets.

These results demonstrate that DTL and DDTL not only excel in accuracy and parameter-efficiency, but also offer strong task generalization across diverse domains. In particular, DDTL's performance on structure-sensitive tasks suggests enhanced disentanglement capability, making it a compelling choice for real-world deployment in resource-constrained and multi-task scenarios.

Table 2. Comparison of average accuracy (%) between DTL and DDTL across ViT models of different sizes on VTAB-1K. DDTL consistently outperforms DTL, with more notable gains in smaller models.

Method	ViT-B	ViT-S	ViT-T
DTL	77.44	75.37	70.97
DDTL	**77.52**	**76.02**	**71.85**

Different model scales. To further understand the performance dynamics of DDTL, we analyze its accuracy across ViT models of varying sizes, as reported in Table 2. The results demonstrate that DDTL consistently outperforms DTL at all model scales, highlighting the general applicability of our dual-side design.

Notably, the improvements are more significant in smaller models: DDTL surpasses DTL by 0.88 percentage points on ViT-Tiny and by 0.65 points on ViT-Small. These gains indicate that DDTL is particularly beneficial for models with limited representational capacity, as the additional fine-grained adaptations from both pre- and post-attention features help compensate for their architectural constraints.

In contrast, the performance gain on ViT-Base is marginal (+0.08%), suggesting that larger models may already possess sufficient adaptation capacity, and the benefits of dual CSNs are less pronounced. This observation aligns with the intuition that capacity-limited models stand to benefit the most from enhanced parameter efficiency and deeper feature disentanglement.

Overall, this analysis confirms the effectiveness of DDTL, particularly in improving transfer performance for small- and mid-sized vision models, making it a compelling choice for efficient model deployment in resource-constrained environments.

Table 3. Accuracy (%) of DTL and DDTL under two block dropping strategies: loss-guided (top) and cosine similarity (bottom), with 0 to 5 blocks dropped. DDTL consistently outperforms DTL, and cosine similarity yields better results when fewer blocks are removed.

Method	**Drop-0**	**Drop-1**	**Drop-2**	**Drop-3**	**Drop-4**	**Drop-5**
DTL (Loss-guided)	71.0	70.4	69.6	68.2	66.1	64.6
DDTL (Loss-guided)	71.8	71.5	70.8	69.1	68.0	66.7
DTL (Cosine-sim)	71.0	70.6	70.2	67.7	64.6	62.5
DDTL (Cosine-sim)	71.8	71.8	71.3	69.4	66.5	64.2

Block Dropping Experiment We compare two block dropping strategies, loss-guided and cosine similarity, under both DTL and DDTL settings. As shown in Table 3, DDTL consistently achieves higher accuracy than DTL across all drop levels and both strategies, demonstrating stronger robustness to structural pruning.

Cosine similarity performs better than loss-guided when only 1–2 blocks are removed. In the case of DDTL, it maintains 71.8% accuracy even after dropping one block, which is higher than the original DTL baseline. Notably, the cosine similarity strategy does not require evaluating each block's individual impact on performance, making it significantly more efficient. This allows for quick, lightweight pruning during training.

In practice, we recommend using cosine similarity to drop 1–2 redundant blocks, which achieves meaningful compression without compromising accuracy. For more aggressive pruning, the loss-guided strategy offers greater stability and better accuracy retention.

4.3 Ablation Experiment

Effect of M on Accuracy and Memory. We investigated how the addition starting point M of the side network affects the performance and efficiency of the model . As shown in Figure 2, increasing M consistently reduces GPU memory

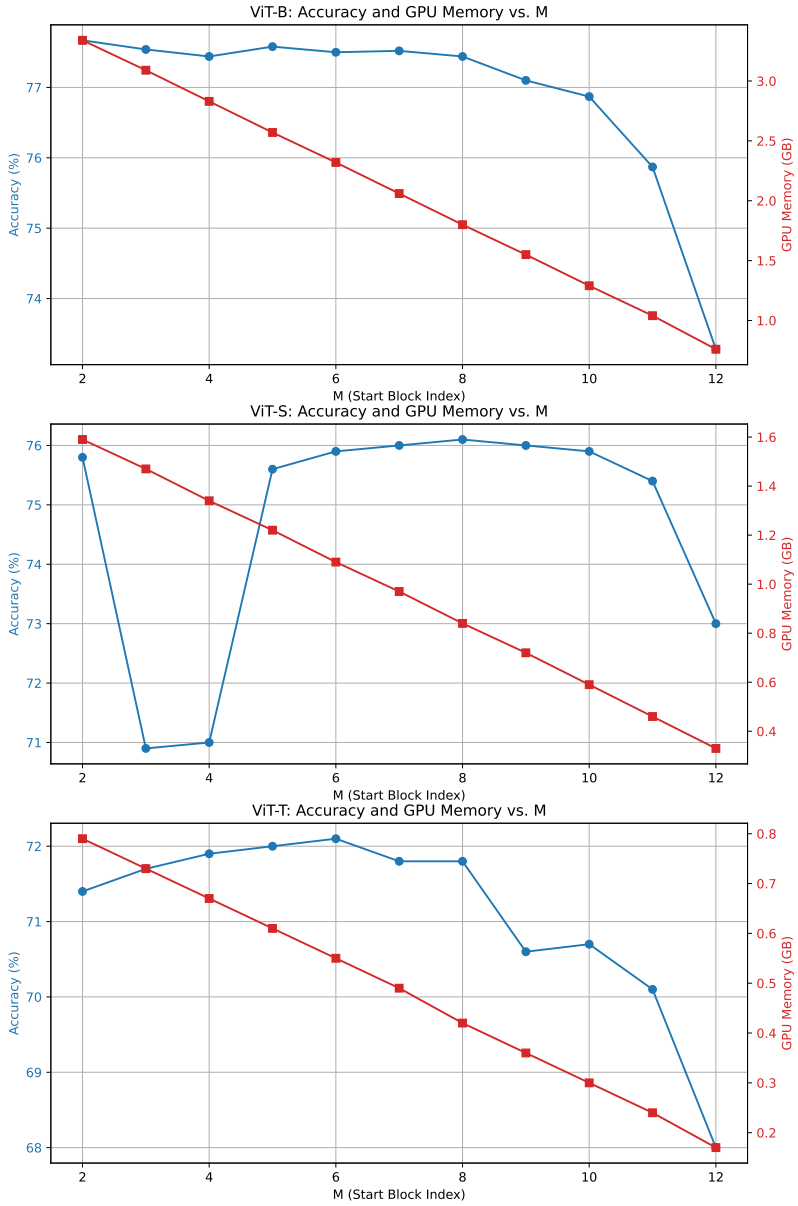

Fig. 2. Accuracy and GPU memory usage of DDTL under different M values on ViT-T, ViT-S, and ViT-B. A smaller M introduces side networks earlier, resulting in better accuracy but higher memory cost. The trade-off curve reveals that mid-range M values (e.g., $M = 5$ to 7) often achieve the best balance between performance and efficiency.

usage, but may also leads to a gradual drop in accuracy, highlighting the trade-off between performance and resource savings.

For ViT-B, accuracy remains relatively stable from $M = 2$ to $M = 7$, while GPU usage is nearly halved, suggesting that moderate M values (e.g., 6 or 7) offer a favorable trade-off. ViT-S peaks at $M = 8$, and ViT-T achieves its best accuracy around $M = 5$ or $M = 6$. In all cases, larger M values lead to significant accuracy loss.

These results confirm that the optimal M depends on model size. Smaller models benefit more from mid-to-late adaptation, while larger models can tolerate earlier intervention. The tunable design of DDTL enhances its flexibility under memory and latency constraints.

Ablation on Side Network Placement. To investigate the role of side network placement, we compare three variants: DTL (side networks before each block), DTL-a (side networks only after the attention module), and DDTL (dual placement before and after). As summarized in Table 4, DTL and DTL-a share the same parameter count (0.05M) and memory usage (1.8 GB), but DTL achieves better accuracy (77.4% vs. 77.1%), indicating that injecting task-specific features before block processing is more effective than doing so after attention.

DDTL further improves performance to 77.5% with only a small increase in parameters and memory, highlighting the complementary effect of combining pre- and post-attention adaptations. These results validate the design choice in DDTL and show that dual-side injection leads to more robust knowledge transfer.

Table 4. Ablation study comparing DTL, DTL-a (single side after attention), and DDTL on ViT-B. DDTL achieves the best accuracy with slightly higher GPU usage, while DTL-a underperforms, indicating the importance of dual-side adaptation.

Method	Params (M)	GPU Mem. (GB)	Accuracy (%)
DTL	0.05	1.80	77.4
DTL-a	0.05	1.80	77.1
DDTL	0.09	2.00	**77.5**

5 Conclusion

In this work, we proposed **DDTL**, a novel dual-branch extension to Disentangled Transfer Learning that introduces compact side networks before each block and after the attention modul. This design enables more granular and effective feature adaptation while preserving the backbone structure. Through extensive experiments on VTAB-1K and across ViT models of various scales, we showed

that DDTL consistently outperforms existing PETL baselines, particularly on small and mid-sized models, with low parameter and memory overhead.

We further explored strategies for efficient deployment, including two block dropping strategies. Our analysis demonstrates that cosine similarity-based pruning can reduce model depth by 1–2 blocks with acceptable accuracy loss, offering a practical trade-off between compression and performance. DDTL's flexibility, robustness, and efficiency make it a promising PETL solution for scalable vision model transfer in memory-constrained environments.

Acknowledgments. This research was partly supported by the National Natural Science Foundation of China under Grant 62276123.

References

1. Chen, S., Ge, C., Tong, Z., Wang, J., Song, Y., Wang, J., Luo, P.: Adaptformer: Adapting vision transformers for scalable visual recognition. Advances in Neural Information Processing Systems **35**, 16664–16678 (2022)
2. Dettmers, T., Pagnoni, A., Holtzman, A., Zettlemoyer, L.: Qlora: Efficient finetuning of quantized llms. Advances in neural information processing systems **36**, 10088–10115 (2023)
3. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al.: An image is worth 16x16 words: Transformers for image recognition at scale (2020), https://arxiv.org/abs/2010.11929
4. Du, S., Zhang, G., Wang, K., Wang, Y., Yue, H., Zhang, G., Ding, E., Wang, J., Xu, Z., Yuan, C.: Alore: Efficient visual adaptation via aggregating low rank experts (2024), https://arxiv.org/abs/2412.08341
5. Fu, M., Zhu, K., Wu, J.: Dtl: Disentangled transfer learning for visual recognition. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 38, pp. 12082–12090 (2024)
6. He, K., Chen, X., Xie, S., Li, Y., Dollár, P., Girshick, R.: Masked autoencoders are scalable vision learners. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 16000–16009 (2022)
7. Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., Attariyan, M., Gelly, S.: Parameter-efficient transfer learning for nlp. In: International conference on machine learning. pp. 2790–2799. PMLR (2019)
8. Hu, E.J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W., et al.: Lora: Low-rank adaptation of large language models. ICLR **1**(2), 3 (2022)
9. Jia, M., Tang, L., Chen, B.C., Cardie, C., Belongie, S., Hariharan, B., Lim, S.N.: Visual prompt tuning. In: European conference on computer vision. pp. 709–727. Springer (2022)
10. Jie, S., Deng, Z.H.: Fact: Factor-tuning for lightweight adaptation on vision transformer. In: Proceedings of the AAAI conference on artificial intelligence. vol. 37, pp. 1060–1068 (2023)
11. Jin, T., Du, E., Wang, C., Xu, W., Luo, D.: Unifying structure and activation: A comprehensive approach of parameter and memory efficient transfer learning (2025), https://arxiv.org/abs/2503.08154

12. Lester, B., Al-Rfou, R., Constant, N.: The power of scale for parameter-efficient prompt tuning (2021), https://arxiv.org/abs/2104.08691
13. Li, X.L., Liang, P.: Prefix-tuning: Optimizing continuous prompts for generation (2021), https://arxiv.org/abs/2101.00190
14. Lian, D., Zhou, D., Feng, J., Wang, X.: Scaling & shifting your features: A new baseline for efficient model tuning. Advances in Neural Information Processing Systems **35**, 109–123 (2022)
15. Liu, X., Ji, K., Fu, Y., Tam, W.L., Du, Z., Yang, Z., Tang, J.: P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks (2021), https://arxiv.org/abs/2110.07602
16. Liu, Z., Hu, H., Lin, Y., Yao, Z., Xie, Z., Wei, Y., Ning, J., Cao, Y., Zhang, Z., Dong, L., et al.: Swin transformer v2: Scaling up capacity and resolution. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 12009–12019 (2022)
17. Luo, G., Huang, M., Zhou, Y., Sun, X., Jiang, G., Wang, Z., Ji, R.: Towards efficient visual adaption via structural re-parameterization (2023), https://arxiv.org/abs/2302.08106
18. Sung, Y.L., Cho, J., Bansal, M.: Lst: Ladder side-tuning for parameter and memory efficient transfer learning. Advances in Neural Information Processing Systems **35**, 12991–13005 (2022)
19. Zaken, E.B., Ravfogel, S., Goldberg, Y.: Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models (2021), https://arxiv.org/abs/2106.10199
20. Zhai, X., Puigcerver, J., Kolesnikov, A., Ruyssen, P., Riquelme, C., Lucic, M., Djolonga, J., Pinto, A.S., Neumann, M., Dosovitskiy, A., et al.: A large-scale study of representation learning with the visual task adaptation benchmark (2019), https://arxiv.org/abs/1910.04867
21. Zhang, J.O., Sax, A., Zamir, A., Guibas, L., Malik, J.: Side-tuning: a baseline for network adaptation via additive side networks. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16. pp. 698–714. Springer (2020)
22. Zhang, Y., Zhou, K., Liu, Z.: Neural prompt search. IEEE Transactions on Pattern Analysis and Machine Intelligence (2024)
23. Zhang, Y., Chen, H., Zhang, X., Chu, X., Song, L.: Dyn-adapter: Towards disentangled representation for efficient visual recognition. In: European Conference on Computer Vision. pp. 433–448. Springer (2024)