TADFormer: Task-Adaptive Dynamic TransFormer for Efficient Multi-Task Learning

Supplementary Material

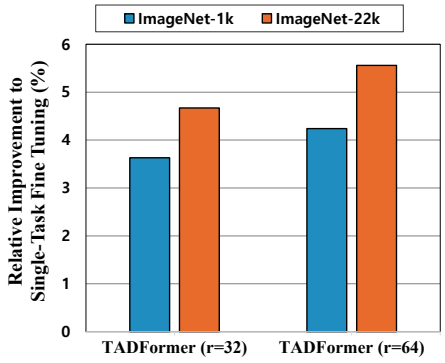
In the supplementary material, we provide more comprehensive results as follows.

- More results using other backbone, pretraining dataset, and decoders
- Analysis on computational efficiency
- TADFormer on adapter-based methods

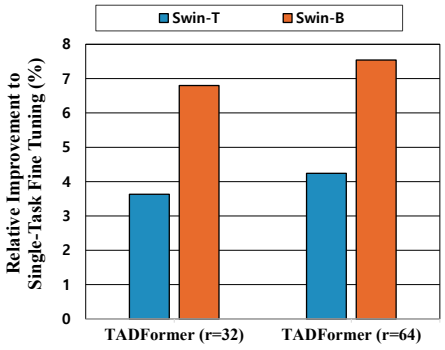
6. More Results

6.1. TADFormer using other Backbone and Pretraining Dataset

Extending the experiment in Fig. 8, we evaluated the performance of TADFormer using larger pretraining dataset and backbone. As shown in Fig. 9, using the larger pretraining dataset improves performance significantly. Similarly, using the larger backbone, such as Swin-B, also resulted in improved performance. It is worth of noting that the higher relative improvement is achieved when the larger back (Swin-B) or training dataset (ImageNet-22k) are used. These results support the effectiveness of our approach, considering that the performance of single-task fine-tuning, which was used to compute the relative performance improvement, also improves. This demonstrates that TADFormer is the scalable method that can effectively adapt to backbones and pretraining datasets of varying sizes.



(a) ImageNet-1k vs ImageNet-22k



(b) Swin-T vs Swin-B

Figure 9. (a) illustrates the performance difference when using Swin-T pretrained on ImageNet-1k and ImageNet-22k as the backbone of TADFormer. (b) illustrates the performance variation of TADFormer when employing Swin-T and Swin-B, both pretrained on ImageNet-1k.

6.2. TADFormer using Other Decoders

We further evaluated the performance of TADFormer in combination with other decoders. We also compared it with the performance of MTLoRA when using the same decoder. Various decoders commonly used in dense prediction tasks were adopted, including HRNet [42], SegFormer [47], and Atrous Spatial Pyramid Pooling (ASPP) [6]. The Swin-T pretrained on ImageNet-22k was used as the encoder. As shown in Table 3, TADFormer demontrates superior multi-task learning performance with fewer trainable parameters compared to MTLoRA across all decoder configurations, confirming that our method is flexible and can be integrated with various decoder architectures. Additionally, ASPP shows the best performance as the decoder with the largest number of trainable parameters, indicating that the choice of decoder enables for an effective trade-off between the performance and the number of trainable parameters.

7. Analysis on Computational Efficiency

Fig. 10 shows the analysis on the computational efficiency in terms of GFLOPs and the number of trainable parameters with respect to the number of tasks. The GFLOPs for both MTLoRA and TADFormer are directly measeured in this experiment. Compared to MTLoRA [1], TADFormer requires slightly more GFLOPs but significantly fewer trainable parameters. This is because the TCP-operator in TADFormer does not require additional parameters, but instead needs additional computation for extracting task-adapted features, and DTF requires additional operations for parameter generation. While TADFormer has a marginal increase in GFLOPs, the substantial reduction in trainable parameters demonstrates its scalability and suitability for efficient multi-task learning scenarios, especially as the number of tasks increases. This trade-off indicates the efficiency of TADFormer in balancing training complexity and parameter optimization.

Table 3. **Performance comparison with other decoders:** For the encoder, we use TADFormer ($r = 32$) and MTLoRA ($r = 32$) with the Swin-T backbone pretrained on ImageNet-22k.

Method	Model Encoder	Decoder	SemSeg ($mIoU \uparrow$)	Human Parts ($mIoU \uparrow$)	Saliency ($mIoU \uparrow$)	Normals ($rmse \downarrow$)	$\Delta m(\%)$	Trainable Param. (M) Decoder / All
MTLoRA [1]	Swin-T	HRNet [42]	69.44	61.08	63.24	16.47	+2.93	1.94 / 6.08
		SegFormer [47]	69.59	61.13	63.74	16.62	+3.00	2.08 / 6.22
		ASPP [6]	72.32	60.98	63.04	16.51	+3.83	12.44 / 16.58
TADFormer	Swin-T	HRNet [42]	72.05	61.6	65.45	16.7	+4.67	1.94 / 4.78
		SegFormer [47]	72.33	61.16	65.8	16.87	+4.51	2.08 / 4.91
		ASPP [6]	73.66	60.37	65.27	16.43	+5.09	12.44 / 15.27

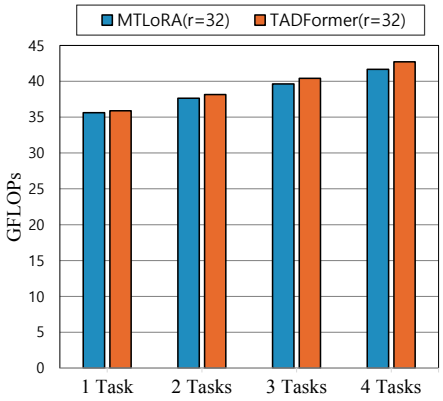
Table 4. **Performance comparison with Adapter-based PEFT Methods:** In the adapter-based PEFT, the input feature is down-projected and up-projected within the adapter ($d \rightarrow r \rightarrow$ d), where d is the dimension of an input feature and r is the dimension of hidden layer, which is also called rank. In our experiments, we consider two types of projection dimensions: 1) $\rho = \frac{d}{r}$ denotes the down-projection ratio used in the adapter, 2) $r = 64$ denotes a fixed down-projected channel dimension. Additionally, 'seq' and 'par' indicate the sequential and parallel configurations of the adapter with MLP module as shown in Fig. 11. This experiment demonstrates the performance of integrating TADFormer with two adapter-based PEFT methods: AdaptFormer and VMT-Adapter. AdaptFormer uses a shared adapter structure for all tasks. VMT-Adapter, similar to AdaptFormer, utilizes a shared adapter but additionally incorporates task-specific scaling and shift operations. $*$ indicates that the results were reproduced by our implementation, as there is no code available. All results were obtained using the Swin-T pre-trained on ImageNet-1k as in Table 1.

Index	Method	SemSeg ($mIoU \uparrow$)	Human Parts ($mIoU \uparrow$)	Saliency ($mIoU \uparrow$)	Normals ($rmse \downarrow$)	$\Delta m(\%)$	Trainable Parameters (M)
S	Single Task	67.21	61.93	62.35	17.97	0	112.62
M1	MTL - Tuning Decoders Only	65.09	53.48	57.46	20.69	-9.95	1.94
M2	MTL - Full Fine Tuning	67.56	60.24	65.21	16.64	+2.23	30.06
A1	AdaptFormer (seq) [7] ($\rho = 4$)	69.01	58.2031	63.545	18.1676	-0.63	3.64
A2	AdaptFormer (par) [7] ($\rho = 4$)	55.28	50.63	60.51	18.55	-10.54	3.64
A3	AdaptFormer (seq) [7] ($r = 64$)	68.84	57.84	63.57	18.46	-1.23	3.12
A4	AdaptFormer (par) [7] ($r = 64$)	55.18	50.39	60.36	18.78	-11.06	3.12
AO1	AdaptFormer+Ours (seq) ($\rho = 4$)	72	59.62	64.94	17.4	2.69	4.48
AO2	AdaptFormer+Ours (par) ($\rho = 4$)	61.41	52.93	62.88	17.57	-5.02	4.48
AO3	AdaptFormer+Ours (seq) ($r = 64$)	71.74	58.56	64.38	17.52	1.76	3.67
AO4	AdaptFormer+Ours (par) ($r = 64$)	60.37	52.18	62.46	17.83	-6.25	3.67
V1	VMT-Adapter (seq)* [49] ($\rho = 4$)	68.98	58.44	63.43	18.26	-0.71	3.65
V2	VMT-Adapter (par)* [49] ($\rho = 4$)	55.4	50.98	60.45	18.5	-10.32	3.65
V3	VMT-Adapter (seq)* [49] ($r = 64$)	68.8	58	63.59	18.42	-1.12	3.14
V4	VMT-Adapter (par)* [49] ($r = 64$)	55.25	50.32	60.38	18.76	-11.03	3.14
VO1	VMT-Adapter+Ours (seq) ($\rho = 4$)	71.91	59.6	64.7	17.37	+2.59	4.49
VO2	VMT-Adapter+Ours (par) ($\rho = 4$)	60.89	52.59	62.58	17.57	-5.48	4.49
VO3	VMT-Adapter+Ours (seq) ($r = 64$)	71.7	58.72	64.64	17.57	+1.85	3.68
VO4	VMT-Adapter+Ours (par) ($r = 64$)	59.81	51.55	62.3	17.9	-6.87	3.68

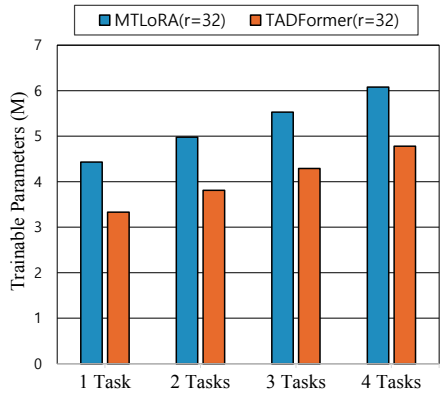
8. TADFormer on Adapter-Based Methods

Experimental Setup. To analyze the extensibility of TADFormer into other PEFT methods, we experimented TADFormer with adapter-based methods [7, 49]. The comparative analysis is shown in Table 4. The first column (Index) identifies the structures of the modules used in the experiments. The AdaptFormer [7] is the adapter-based PEFT method for single task (**A1**−**A4**). The VMT-Adapter [49] extends the Adapter for MTL in a way that employs the task-shared adapter, similar to AdaptFormer [7], while extracting task-specific features through task-wise scaling and shift operations (**V1**−**V4**). The experiments of the VMT-Adapter were conducted by our implementation, as there is no code available. We also implemented our method on the AdaptFormer (**AO1**−**AO4**) and the VMT-Adapter (**VO1**−**VO4**). 'seq' and 'par' indicate the sequential and parallel configurations of the adapter with MLP module as shown in Fig. 11. Please refer to the caption of Table 4 for more details on ρ and r.

An example of applying our method, TADFormer, to the AdaptFormer [7] is described in Fig. 12 (**AO2** or **AO4**). This design allows the TADFormer module to be integrated into adapter-based PEFT methods, taking into account both



(a) Comparison of GFLOPs between MTLoRA ($r = 32$) and TADFormer ($r = 32$) with respect to the number of tasks.



(b) Comparison of the number of trainable parameters between MTLoRA ($r = 32$) and TADFormer ($r = 32$) with respect to the number of tasks.

Figure 10. **Efficiency of TADFormer with different number of tasks**: The experiments were run at the rank $r = 32$.

task and input contexts. This architecture can be applied to both parallel and sequential configuration of adapters and is equally applicable to other adapter-based method such as VMT-Adapter [49].

In the following, we compared the existing adapter-based methods [7, 49] with our method implemented on the adapter framework. For a fair comparison, we evaluated the performance for the cases with the similar amount of trainable parameters, though the results of all possible combinations are provided in Table 4. Our code on the adapter-based experiments is submitted as supplementary material.

AdaptFormer vs. AdaptFormer with Ours. For a fair comparison in terms of the number of trainable parameters, we compared **A1, A2** and **AO3, AO4** that have comparable numbers of trainable parameters. In both comparison, TADFormer demonstrated higher Δm values when integrated into both sequential and parallel configurations. This reveals that the application of TADFormer to AdaptFormer results in an overall enhancement in MTL performance.

VMT-Adapter vs. AdaptFormer with Ours. As the VMT-Adapter [49] is an extension of the AdaptFormer [7], we applied our method to the AdaptFormer, and then compared it with VMT-Adapter. To be specific, the model proposed in [49] uses the down-projection ratio of $\rho = 4$ (**V2**). For a fair comparison in terms of the number of trainable parameters, we compared it with the AdaptFormer combined with TADFormer using $r = 64$ (**AO4**). This makes their number of parameters comparable. In comparison, **AO4** achieves a Δm increase of 4.07 with only an additional 0.02 M trainable parameters compared to **V2**. The comparison of **V1** and **AO3** also shows a similar tendency. This result demonstrates that the structure of TADFormer is more effective for multi-task learning than the scaling and shift operations used in the VMT-Adapter.

VMT-Adapter vs. VMT-Adapter with Ours. We also compared **V1**, **V2** with **VO3**, **VO4**, which have comparable numbers of trainable parameters, demonstrating that the performance has notably improved with only 0.3M increase in trainable parameters. This suggests that even when the adapter's down-projection channel dimension is reduced to reduce trainable parameters, the structure of TADFormer is capable of efficiently and effectively extracting multi-task representations.

To sum up, these results confirm that the TADFormer can be successfully integrated into various adapter-based methods and is the scalable multi-task PEFT method, compatible with both LoRA and adapter-based frameworks.

Table 5. **Performance comparison with fully-tuned MTL models:** The fully-tuned MTL models (Swin-B) and ours (Swin-L) use different backbones, as these SOTA models need more parameters on the task decoders.

Method	SemSeg ($mIoU \uparrow$)	Human Parts ($mIoU \uparrow$)	Saliency ($maxF \uparrow$)	Normals ($rmse \downarrow$)	Entire Params (M)	Trainable Params (M)
Taskprompter (Swin)	78.82	65.68	84.73	14.26	218	218
MLoRE (Swin)	79.97	68.19	84.89	14.42	259	259
TADFormer (Swin)	**78.03**	**69.25**	78.68	16.09	219	24

Table 6. **Additional ablation study on TADFormer**

Method	SemSeg ($mIoU \uparrow$)	Human Parts ($mIoU \uparrow$)	Saliency ($mIoU \uparrow$)	Normals ($rmse \downarrow$)	$\Delta m(\%)$	Trainable Params (M)
TADFormer (r=32)	70.2	60	65.71	16.57	3.63	4.78
Without Gating	70.56	59.72	65.41	16.78	3.23	4.78
Tuning PM	70.39	60.01	65.47	16.67	3.47	6.32

9. Significance of PEFT-based MTL methods

As the model size grows, training large-scale models for downstream tasks has become resource-intensive even with modern GPUs. Consequently, PEFT has been actively studied to reduce training complexity, which is especially beneficial in more complex MTL models where multiple tasks run simultaneously. To clarify the performance reported

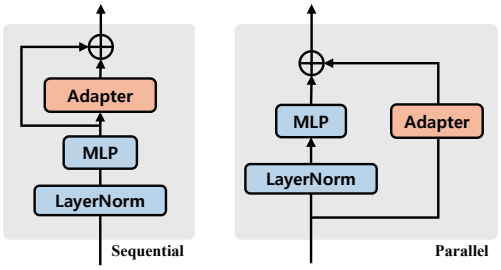


Figure 11. Adapter configuration: Both sequential and parallel configurations are possible in adapter-based PEFT framework [7].

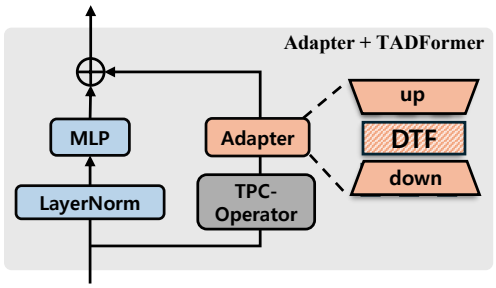


Figure 12. Overview of Adapter + TADFormer in parallel adapter configuration

Table 7. **Results for rank=4, 8, and 16**

Method	SemSeg ($mIoU \uparrow$)	Human Parts ($mIoU \uparrow$)	Saliency ($mIoU \uparrow$)	Normals ($rmse \downarrow$)	$\Delta m(\%)$	Trainable Params (M)
MTLoRA+ (r=4)	68.12	57.77	63.14	17.6	-0.52	2.57
MTLoRA+ (r=8)	68.54	58.3	63.57	17.41	+0.29	3.15
MTLoRA+ (r=16)	68.28	58.7	64.323	17.034	+1.19	4.29
TADFormer (r=4)	69.22	57.97	63.77	17.54	+0.31	2.68
TADFormer (r=8)	69.46	58.42	64.22	17.32	+1.07	2.97
TADFormer (r=16)	69.79	59.27	65.04	16.91	+2.44	3.56

in Table 1, we compare our method with fully-tuned MTL models. The fully-tuned MTL models use a larger backbone (ViT-L), while ours uses Swin-T. An evaluation was conducted with a similar number of parameters as shown in Table 5, indicating that ours achieves comparable performance only with 24M trainable parameters.

10. Additional ablation study on TADFormer

We conducted additional ablation studies on TADFormer. As shown in Table 6, removing the stage-wise gating module (Sec. 3.5) led to a performance drop of 0.4%. Additionally, fine-tuning the patch merging (PM) module (Sec. 3.6) does not lead to performance gain, while increasing the number of trainable parameters (4.78M $\rightarrow$ 6.32M).

We experimented with different ranks in TADFormer. As shown in Table 7, TADFormer consistently outperforms MTLoRA+ [1] with fewer or a similar number of trainable parameters.