MTLoRA: A Low-Rank Adaptation Approach for Efficient Multi-Task Learning

Supplementary Material

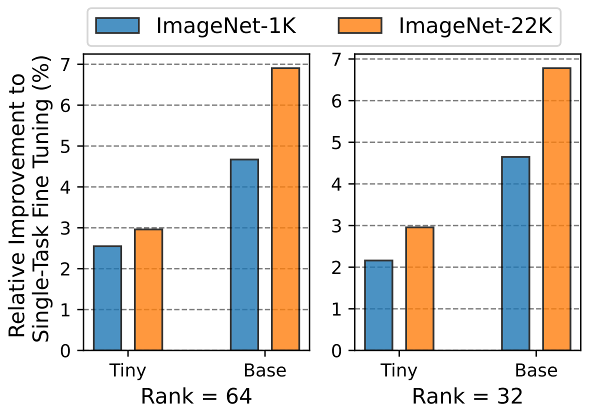

Figure 1. Performance with MTLoRA with different pretraining datasets on various Swin-Transformer backbones.

1. Different pretraining datasets: ImageNet-1K vs ImageNet-22K

We analyze the effect of using different pretraining datasets on the performance of MTLoRA. Figure 1 shows the relative improvement in accuracy compared to the single task models of MTLoRA when applied to Swin-Tiny and Swin-Base pretrained on ImageNet-$1K$ and ImageNet-$22K$. The figure shows that the pretraining dataset can have a significant impact on the model's performance. Specifically, using a model pre-trained on a richer dataset (i.e., ImageNet-22K) results in a better performance on the downstream tasks without any additional cost on our parameter-efficient training methodology.

2. MTLoRA with different Adaptation Scales

The scale value α in Equation 2 determines how much the fine-tuned model can deviate from the original baseline model. For example, A scale value of 0 is the same as not using the LoRA weights and only using the base model weights, and a scale value of 1 means that both the LoRA weights as well as the base model weights have the same influence. It is common to use α in $1, 2$ for language models [3]; however, we experiment with different scales to analyze their effect when fine-tuning vision transformers for multiple tasks. Figure 2 shows the overall accuracy at different scales. We found that empirically, scale 4 performs the best for the purpose of MTLoRA.

3. MTLoRA with different Backbones

To ensure the generalizability of MTLoRA, we apply it to another vision transformer backbone - Pyramid Vision Transformer [8]. We analyze the impact of using MTLoRA

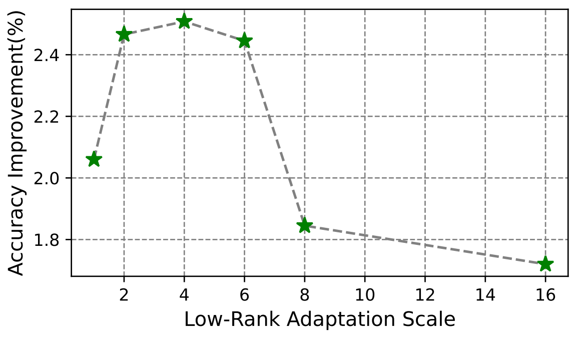

Figure 2. Effect of the hyper-parameter α on the accuracy of MTLoRA on the downstream tasks.

when applied on *PVT-Small*. Table 1 shows that applying MTLoRA offers Pareto-optimal accuracy-efficiency trade-off compared to state-of-the-art paramter-efficient training techniques. Specifically, MTLoRA achieves higher accuracy (Δm) when compared to Hyperformer [6] while training $2\times$ fewer parameters.

4. MTLoRA with different decoders

We analyze the effect of using different decoders on the performance of MTLoRA. We choose 3 commonly used decoders for dense prediction tasks: (1) HRNet [7], which interpolates and concatenates multi-scale features from the hierarchical backbone and then passes them to a couple of MLP layers. (2) SegFormer [9], which uses MLP layers to combine the multi-scale features from the hierarchical backbone, then parse them to one last MLP layer for prediction. (3) Atrous Spatial Pyramid Pooling (ASPP) [1], which has a more complicated architecture utilizing spatial pyramid pooling capable of extracting multi-scale contextual information by probing the incoming features with filters or pooling operations at multiple rates and multiple effective fields-of-view. We plug those decoders into a pretrained Swin-Tiny backbone, then use our MTLoRA technique to train the model to perform multiple downstream tasks. Table 2 shows that MTLoRA generalizes perfectly when different decoders are used. Different decoders can provide different accuracy-efficiency trade-offs, which provide flexibility to adapt the training budget depending on the application requirements and the available resources.

5. MTLoRA with different number of tasks

In this section, we evaluate MTLoRA in a setting with increased number of tasks. Table 4 demonstrates the accuracy-efficiency trade-off with increasing task numbers.

Table 1. *MTLoRA* versus SoTA parameter-efficient training methods when applied on Pyramid Vision Transformer [8] backbone. The table summarizes the number of trainable parameters in each method. It also includes the accuracy of the downstream tasks as well as the average MTL model's accuracy (Δm).

Method	**SemSeg** ($mIoU \uparrow$)	**Human Parts** ($mIoU \uparrow$)	**Saliency** ($mIoU \uparrow$)	**Normals** ($rmse \downarrow$)	$\Delta m(\%)$	**Trainable Parameters** (M)
Single Task	68.81	61.27	62.67	17.55	0.00	97.51
MTL - Fine Tune Decoders	64.86	51.18	61.54	19.55	-8.85	2.11
Compactor++ [4]	70.29	54.80	63.16	18.82	-3.71	2.20
BitFit [10]	71.41	55.71	64.08	18.69	-2.38	2.34
LoRA [3]	71.89	56.9	64.27	18.48	-1.35	2.41
Adapter [2]	71.94	56.38	64.16	18.75	-1.97	2.90
Polyhistor [5]	71.00	57.52	65.83	17.83	+0.13	7.32
Hyperformer [6]	70.81	57.76	65.49	17.75	+0.14	16.14
MTLoRA ($r = 64$)	69.74	58.08	65.62	17.35	**+1.2**	8.69

Table 2. Evaluating MTLoRA with $rank = 32$ when applied on Swin-Tiny backbone pretrained on ImageNet-22K dataset and various decoders for the downstream dense prediction tasks. The table summarizes the number of parameters for each decoder as well as the total number of parameters. It also includes the accuracy of the downstream tasks as well as the average MTL model's accuracy (Δm).

Decoder	**SemSeg** ($mIoU \uparrow$)	**Human Parts** ($mIoU \uparrow$)	**Saliency** ($mIoU \uparrow$)	**Normals** ($rmse \downarrow$)	$\Delta m(\%)$	**Trainable Parameters** (M) $Decoder/All$
HR-Net [7]	69.44	61.08	63.24	16.47	+2.93	1.94 / 6.08
SegFormer [9]	69.59	61.13	63.74	16.62	+3.00	2.08 / 6.22
ASPP [1]	72.32	60.98	63.04	16.51	+3.83	12.44 / 16.58

Table 3. Analyzing the number of FLOPs for different numbers of tasks for *Individual Task-Specific Adaptation* versus our *Shared Multi-Task Adaptation* depicted in Figures 2(a) and 2(b) respectively. We can clearly notice that our approach is significantly more efficient as the number of tasks increase.

	Individual Task-Specific Adaptation (Figure 2(a))	**Shared Multi-Task Adaptation** (Figure 2(b))
1 Tasks (GFLOPs)	18.48	18.48
2 Tasks (GFLOPs)	36.91	19.50
3 Tasks (GFLOPs)	55.32	20.49
4 Tasks (GFLOPs)	73.74	21.49

The table shows that integrating additional tasks into MTLoRA incurs minimal latency (in terms of trainable parameters) relative to single-task learning and conventional full MTL fine-tuning, while still maintaining superior accuracy compared to these approaches.

6. FLOPs overhead for different number of tasks.

Thank you for the suggestion. As you mentioned, while adding tasks incurs a slight overhead in backbone inference, this cost is marginal as it avoids divergent paths unlike previous techniques. Moreover, these tasks can be batched for efficiency, similar to SWIN's window-based computations, due to their low-rank nature. We've included Table 3 to compare FLOPs against task numbers in *Individual Task-Specific Adaptation* and our *Shared Multi-Task Adaptation* (Figures 2(a) and 2(b)). This highlights the minimal impact on FLOPs in our approach compared to traditional *Individual Task-Specific Adaptation*.

Table 4. Evaluating MTLoRA for **different numbers of tasks**.

Method	$\Delta m(\%)$	**Trainable Parameters** (M)
Single Task (All 4 tasks)	0	112.62
Full MTL Fine-Tuning (All 4 tasks)	+2.23	30.06
MTLoRA (SemSeg and Normals)	+8.7	5.83
MTLoRA (SemSeg and Sal)	+5.2	5.83
MTLoRA (Semseg, Normals, and Sal)	+4.37	6.45
MTLoRA (All 4 tasks)	+2.55	8.34

References

[1] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818, 2018. 1, 2

[2] Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. Towards a unified view of parameter-efficient transfer learning. *arXiv preprint arXiv:2110.04366*, 2021. 2

[3] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen.

Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021. 1, 2

[4] Rabeeh Karimi Mahabadi, James Henderson, and Sebastian Ruder. Compacter: Efficient low-rank hypercomplex adapter layers. *Advances in Neural Information Processing Systems*, 34:1022–1035, 2021. 2

[5] Yen-Cheng Liu, Chih-Yao Ma, Junjiao Tian, Zijian He, and Zsolt Kira. Polyhistor: Parameter-efficient multi-task adaptation for dense vision tasks. *Advances in Neural Information Processing Systems*, 35:36889–36901, 2022. 2

[6] Rabeeh Karimi Mahabadi, Sebastian Ruder, Mostafa Dehghani, and James Henderson. Parameter-efficient multi-task fine-tuning for transformers via shared hypernetworks. *arXiv preprint arXiv:2106.04489*, 2021. 1, 2

[7] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, et al. Deep high-resolution representation learning for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 43(10):3349–3364, 2020. 1, 2

[8] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 568–578, 2021. 1, 2

[9] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M Alvarez, and Ping Luo. Segformer: Simple and efficient design for semantic segmentation with transformers. *Advances in Neural Information Processing Systems*, 34:12077–12090, 2021. 1, 2

[10] Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. *arXiv preprint arXiv:2106.10199*, 2021. 2