FAAR: Efficient Frequency-Aware Multi-Task Fine-Tuning via Automatic Rank Selection

Supplementary Material

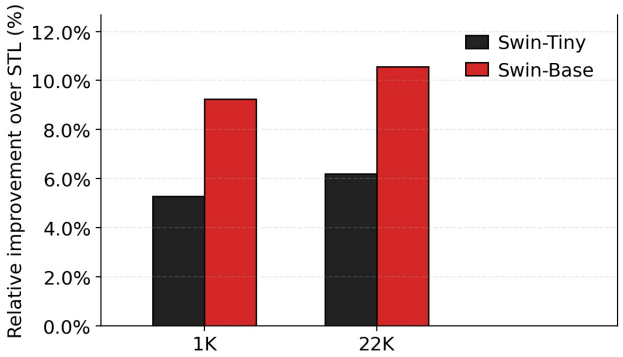

Figure 1. Comparison of performance of FAAR on different ImageNet dataset versions [4] and on two different Swin Transformer configurations [5].

In the supplementary material, we provide a more comprehensive understanding of our method by presenting the following.

- More results on different pre-trained datasets, backbones, and decoders.
- More ablation results that demonstrate the superiority of FAAR.
- Analysis of the computational efficiency gains obtained by FAAR.

1. Comparison of different backbones

Tab. 1 presents results obtained on glowingly larger configurations of swin transformers, namely *Tiny*-, *Base*- and *Large*- which achieve overall MTL performance of $\Delta = +5.28$, $\Delta = +9.23$ and $\Delta = +10.30$, respectively. Moreover, we present an alternative backbone and present results obtained on the Pyramid Vision Transformer (PVT) backbone with comparison to MTLoRA [1]. These results demonstrate both the superiority and scalability of our method.

2. ImageNet 1k vs ImageNet 22k

While the main results of our paper are reported using a backbone pre-trained on ImageNet 1k for ease of comparability, we analyze, in Fig. 1, the effect of using different versions of the ImageNet dataset [4] to pre-train our Swin Transformer backbone. ImageNet-1K is a smaller version of the dataset of approximately 1.28 million whereas ImageNet-22K is a much bigger dataset variant which consists of 11x times more images. Similarly to previous work [1, 2], we observe an improvement in performance of 0.92% and 1.3 % in relative improvement over single-task learning between the tiny and base configuration of our swin transformer for ImageNet-1K and ImageNet-22K, respectively. These results demonstrate the effectiveness and scalability of our method.

Table 1. FAAR Accuracy/Efficiency Comparison

Model	Sem.	H.P	Sal.	Nor.	Δm	B.p	T.p
FAAR - Tiny	72.02	61.25	66.11	16.35	+5.28	1.29	3.38
FAAR - Base	76.80	66.34	**66.24**	16.30	+9.23	3.05	5.14
FAAR - Large	**77.80**	**68.03**	66.14	**16.26**	**+10.30**	6.96	9.05
STL- PVT	68.81	61.27	62.67	17.35	0	95.57	97.51
MTLoRA [2] - PVT	69.74	58.08	65.62	17.35	+1.20	6.75	8.69
FAAR ($\rho = 0.95$) - PVT	**73.28**	**62.20**	**65.95**	**16.79**	**+4.12**	**1.54**	**3.63**

3. Comparison of different decoders

With our frequency-centric decoder, named TS-PD, being central in our work, we provide decoder-based comparison in Tab. 2. Using the same encoder for all methods, a Swin-Tiny, pre-trained on ImageNet-22k. We compare 3 task-specific decoders typically used for dense visual tasks, namely HRNet [7], which is our default decoder used in our main experiments but also SegFormer [9] and Atrous Spatial Pyramid Pooling (ASPP) [3]. For fair comparison, we implement TS-PD inside of each. FAAR shows superior performance with fewer trainable parameters overall. While there are minimal, yet, additional decoder parameters due to the injection of our TS-PD method, the substantial drop in number of parameters brought by our Performance-Driven Rank Shrinking (PDRS) method brings the overall number of parameters below comparable methods. Additionally, similarly to previous work, we observe that the best performance is achieved with ASPP decoders. This demonstrates, to a further extent, the importance of the choice of decoder to tackle the complexity/efficiency trade-off.

Table 2. Performance comparison with different decoders, for various methods, on PASCAL-Context [6]. All models are Swin-Tiny encoders, pre-trained on ImageNet-22k.

Model	Decoder	SemSeg (*mIoU* ↑)	HumanParts (*mIoU* ↑)	Saliency (*mIoU* ↑)	Normals (*rmse* ↓)	Δm (%)	Trainable Params (M) Decoder / All
MTLoRA [1] (r=64)	HRNet [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 [3]	72.32	59.84	60.98	16.51	+3.83	12.44 / 16.58
TADFormer [2] (r=64)	HRNet [7]	72.05	61.60	65.45	16.70	+4.67	1.94 / 4.78
	Segformer [9]	72.33	61.16	65.80	16.87	+4.51	2.08 / 4.91
	ASPP [3]	73.66	60.37	65.27	16.43	+5.09	12.44 / 15.27
FAAR (r_{init}=64)	HRNet [7]	73.30	61.70	66.30	16.25	+6.15	2.09 / 3.38
	Segformer [9]	73.39	61.73	66.35	16.35	+6.08	2.23 / 3.52
	ASPP [3]	74.04	62.09	66.11	16.17	+6.62	12.59 / 13.88

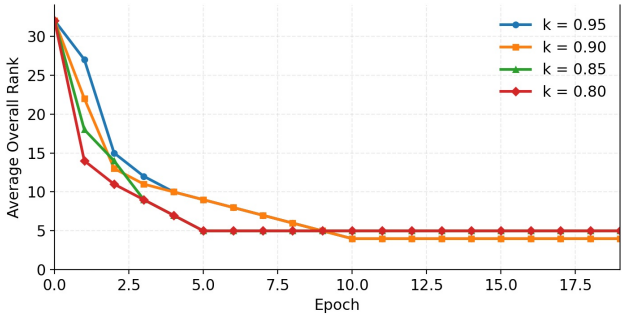

Figure 2. Comparison of different coverage criterions (noted k), for simplicity we keep $\rho_{shared} = \rho_{task}$.

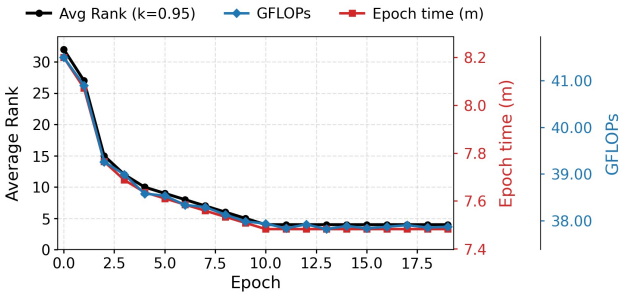

Figure 3. Analysis of the effect of the rank shrinking method on both epoch running time (in minutes), for batch size 32 and GFLOPs.

4. Analysis of FAAR computational efficiency

We analyze in Fig. 3 and Tab. 3 the effect of our method on computational efficiency from both a wall-clock time, time complexity and computational complexity (in GFLOPs) perspective. All the presented illustrations were obtained using a Swin-Tiny as backbone with a batch size 32 for 300 epochs with a r or r_{init} (for our method) of 32. In addition, results were obtained on a single A40 NVIDIA GPU.

Table 3. Comparison of GFLOPs and total training time on Pascal-Context [6]. All configurations include a starting rank of 32. The number of training epochs is set to 300.

Model	GFLOPs	Total Training Time
MTLoRA [1] (r=32)	≈ 41	≈ 40 hours
TADFormer [2] (r=32)	≈ 42	≈ 40 hours
FAAR (r_{init}=32)	≈ 38	≈ 37 hours

4.1. FAAR: Computational Complexity

Fig. 3 demonstrates the computational complexity of FAAR. By keeping the configuration for which we set $\rho_{shared} = \rho_{task} = 0.95$. Our method shows a combined progression of rank, epoch time and GFLOPs. For the first 20 epochs, our method drastically reduces the rank from an initial rank $r_{init} = 32$ to an efficient rank $r_{eff} = 5$. Over the these starting epochs, we observe a drop in GFLOPs from approximately 42 to 38. This demonstrates the effectiveness of our rank shrinking method in bringing a computationally efficient method. Moreover, we illustrate the superiority of our model compared to similar work in Tab. 3. Our model achieves less complexity by having approximately 38 GFLOPs, reducing the count of 5 and 4 from TadFormer [2] and MTLoRA [1], respectively.

4.2. FAAR: Training Time

We demonstrate the effectiveness of FAAR to reduce training time due to its efficient rank shrinking method. We observe in Fig. 3 a decrease from approximately 8.2 minutes of running time per epoch to approximately 7.5 minutes. Moreover, we show the superiority of our method by achieving, compared to other methods, a reduction of 3 hours, from 40 hours for other works [1, 2] to 37 hours for FAAR.

Table 4. Adapter Locations

TS-Blocks	(4)	(3)	(4,3)	(4,3,2,1)
Δm (%)	+2.5	+1.3	+4.40	+5.44

4.3. Effect of different coverage criterion

We analyze in Fig. 2 the effect of our hyper-parameter coverage criterion, noted ρ_{shared} and ρ_{task}. We observe the behavior of 4 different values of ρ, noted k in the figure. As intended, this hyper-parameter controls the convergence of FAAR to lower ranks. A high value will lead to slow convergence, while a lower value brings a steep and fast convergence. We observe that while the average overall rank of the model varies differently for different values of ρ for the first 10 epochs, all values converge approximately to an average global rank of ≈ 5. We observe in our own experiments, not reported in this paper, that the effect of ρ is minimal, but tends to be slightly more performant and stable at higher values (*i.e.* $\rho_{shared}, \rho_{task} = 0.95$) based on validation set performance.

5. Comparison of different cross-task interactions

Our TS-PD leveraging FFTs addresses an important improvement in dense MTL, overlooked by previous PEFT methods, which is to improve the spatial awareness of each task while maintaining cross-task consistency. This challenge cannot be solved by our rank-pruning strategy PDRS; simply, PDRS and TS-PD both focus on improving the efficiency of MTL yet arise from different aspects and are complementary. Next, we demonstrate in Tab. 5 the efficiency of TS-PD compared to well-known cross-task interactions in MTL. Ours provides on-par improvement at about half the cost in parameters (p), suggesting that the XT-Cons inside TS-PD could be replaced by non-frequency cross-task interactions, however, at the cost of additional parameters.

6. Adapted layer comparison

FAAR follows the findings of MTLoRA [1] regarding the type of trained layers. We experiment with different task-specific adaptation locations and demonstrate in Tab. 4 that (1) the effectiveness of adaptation for all layers and (2) deeper layers (3,4) are more performant for adaptation. This is due to deeper layers learning more complex and task-specialised patterns.

Table 5. Cross-Task Interactions comparison

Method	p	Δm (%)
TS-PD	156,000	+5.28
MTI-Net [8]	337,008	+4.90
PAD-Net [10]	333,720	+5.35

References

[1] Ahmed Agiza, Marina Neseem, and Sherief Reda. Mtlora: A low-rank adaptation approach for efficient multi-task learning, 2024. 1, 2, 3

[2] Seungmin Baek, Soyul Lee, Hayeon Jo, Hyesong Choi, and Dongbo Min. Tadformer : Task-adaptive dynamic transformer for efficient multi-task learning, 2025. 1, 2

[3] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs, 2017. 1, 2

[4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. 1

[5] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows, 2021. 1

[6] Roozbeh Mottaghi, Xianjie Chen, Xiaobai Liu, Nam-Gyu Cho, Seong-Whan Lee, Sanja Fidler, Raquel Urtasun, and Alan Yuille. The role of context for object detection and semantic segmentation in the wild. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. 2

[7] Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang. Deep high-resolution representation learning for human pose estimation, 2019. 1, 2

[8] Simon Vandenhende, Stamatios Georgoulis, and Luc Van Gool. Mti-net: Multi-scale task interaction networks for multi-task learning, 2020. 3

[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, 2021. 1, 2

[10] Dan Xu, Wanli Ouyang, Xiaogang Wang, and Nicu Sebe. Pad-net: Multi-tasks guided prediction-and-distillation network for simultaneous depth estimation and scene parsing, 2018. 3