PETAH: Parameter Efficient Task Adaptation for Hybrid Transformers

Supplementary Material

7. Hyperparameters

7.1. Classification

Hyperparameters for the fine-grained classification benchmarks are given in Tab. 5. For Aircraft, DTD, and Food101, we search for the best-performing parameters on the validation set. For each result in Tab. 3, we repeat each experiment 3 times with different random seeds. For the datasets with validation sets (Aircraft, DTD, Food101), we use the best-performing parameters from that dataset's validation run. For the other datasets (CUB, Pets, Cars), we use the parameters with the best average validation performance on Aircraft and DTD, since these two datasets are most similar to the other 3 in terms of size. We also include the dataset statistics in Tab. 6. Note that for Full FT and Attention FT, we include a separate LR factor for the backbone parameters that is multiplied with the base LR which is used for the linear head. Similarly, for LoRA, PETAH, Consolidator, and SSF, we use a factor for the PEFT parameters. As data augmentation, we only use random cropping and horizontal flipping unless stated otherwise. All experiments were done using 8 NVIDIA A100 GPUs each with a batch size of 128, resulting in a total batch size of 1024.

7.2. Detection and Segmentation

Coco Cascade R-CNN: For detection and instance segmentation, we use a Cascade R-CNN with FPN on Coco. The hyperparameters are mostly the same as in [40] but we lower the resolution to 640×480. We train for 12 epochs using a batch size of 8×2. After a 500-step linear warmup, the LR is reduced in epochs 8 and 11. The optimizer is AdamW with a base LR of 0.0001 and weight decay of 0.05.

ADE20K Semantic Segmentation: For semantic segmentation, we train for 40K steps using a batch size of 8×4 with the AdamW optimizer and a base learning rate of 0.0002 and weight decay 0.0001. The evaluation resolution is 2048×512.

8. EF L1 results

The missing results for the EF L1 from Tab. 3 can be found in Tab. 7. Similar to the results from the main paper, PETAH outperforms other adaptation methods. However, due to its small memory and compute footprint, the EF L1 is substantially worse than the EF L3 and ViT-S.

9. PETAH as regularization

Surprisingly, our PETAH approach is able to outperform full fine-tuning (FT) on some datasets despite using less than 1% of total parameters. Effectively, PEFT methods constrain the optimization to a subspace, which could prevent overfitting. This could be especially important on fine-grained classification datasets with fewer samples. However, there is no clear evidence that overfitting causes FT to perform worse than PEFT, since this phenomenon could also have a different cause, *e.g*. easier optimization. To validate our hypothesis and gain further insights into the impact of PEFT on performance, we retrain the EF L7 with PETAH-2 and full FT with explicit regularization on the 5 smaller classification datasets (See Tab. 8). As regularization, we use RandAugment [15] and Dropout [60] before the classification head. First, we note that both approaches can profit from external regularization and heavy data augmentation seems to be especially beneficial in these low-data regimes. Despite a significant gain in performance for full FT (from 84.56% to 86.48% mean accuracy), PETAH-2 remains the best-performing method with a mean accuracy of 86.73% which implies that its inherent regularization effects are well suited for the adaptation of the EfficientFormer to various vision tasks. However, the gap between FT and PETAH-2 narrows from 0.99 to 0.25, implying that overfitting, which is less likely with such strong regularization, is the reason why full FT performs worse than PETAH.

10. Pruning Strategy

The experimental setup for task adaptation is the same as in the previous section. For full fine-tuning and attention-tuning, we preserve the sparsity mask from pre-training and only finetune the non-zero backbone weights. Results can be found in Tab. 9.

Note that for standard PEFT approaches for dense linear layers of the form $Wx + \Delta Wx$, the resulting transformation is another linear transformation defined by the matrix $W + \Delta W$. Thus PEFT methods have a *strictly smaller* capacity than full fine-tuning W since they restrict the optimization to the subspace defined by the specific choice of ΔW. However, if W is a matrix with a sparsity constraint, the transformed matrix $W + \Delta W$ is generally no longer sparse. If we jointly train W with a sparsity constraint and ΔW with a PEFT subspace constraint, the resulting model would have a strictly *larger* capacity than a model with a sparsity constraint without the additional ΔW. In our context, during task adaptation, W is a *fixed* and *sparse* matrix whereas ΔW is a learnable matrix with a subspace constrained defined by the particular PEFT approach. This suggests that applying PEFT to sparse models can enhance the model's capacity beyond what is achievable by simply fine-

tuning the sparse matrices with the same sparsity layout, as done in the full fine-tuning setting.

For the EF L7 with 90% sparsity, we can see that this theoretical increase in capacity translates into practice and results in our PETAH adaptation outperforming all other adaptation methods. While different ViT adaptations should have the same theoretical properties, these do not translate to performance gains since attention and full fine-tuning result in a similar accuracy. In particular, we note that our EF L7 with a 90% sparsity ratio and PETAH-2 achieves a mean accuracy of 84.91% across all tasks at a base model parameter count of 7.9M and 0.45M parameters per task. In comparison, a sparse ViT-B with LoRA will only achieve 84.45% mean accuracy at 8.1M base model parameters and 0.44M per-task parameters. Our results highlight that the combination of PEFT and pruning for the EfficientFormer is a particularly effective way to achieve small and performant models that can easily be adapted to new downstream applications. In particular, our pruned EF L7 with PETAH adaptation is faster, smaller, and achieves better performance than a standard ViT-S with any adaptation strategy, which would be a common choice for a vision backbone in mobile settings.

While the combination of sparsity and low-rank adaptation has been used for attention approximations [8], up to our knowledge, this is the first demonstration of the potential benefits of combining sparse backbones with PEFT task-adaptation techniques, in particular for computer vision with a hybrid transformer-convolutional backbone.

Table 5. Hyperparameters for the fine-grained classification from Tab. 3

Classification	Linear Probing	Full FT Attention FT	LoRA PETAH Consolidator
Batch size		8×128	
Epochs		200 / 100 (Food101)	
Warmup		2	
Optimizer		AdamW	
Scheduler		Cosine	
Learning Rate	{0.005, 0.01, 0.05}	{0.001, 0.005, 0.01}	{0.001, 0.005, 0.01}
Adapter LR factor	-	{0.001, 0.1, 1.0}	{0.1, 1.0, 5.0, 10.0}
Weight decay		{0, 0.0002, 0.002, 0.02}	

Table 6. Dataset Overview

	CUB	**Cars**	**Pets**	**Aircraft**	**DTD**	**Food**
Train Samples	5994	8144	3680	3334	1880	70700
Validation Samples	-	-	-	3333	1880	5050
Test Samples	5794	8041	3669	3333	1880	25250
Num. classes	200	196	37	102	47	101

Table 7. EF L1 fine-grained classification results omitted from Tab. 3.

	Type	**CUB**	**Cars**	**Pets**	**Aircraft**	**DTD**	**Food**	**Mean**	**#Params**
EF L1	Linear-probing	82.74	63.15	90.48	47.03	**71.77**	79.47	72.44	0
	ATTN FT	79.83	81.98	89.87	59.82	70.66	81.52	77.28	1.5M
	Full FT	79.83	81.98	89.87	59.82	70.66	**85.98**	80.17	11.4M
	LoRA ATTN	82.10	77.52	90.70	57.25	71.01	81.42	76.66	0.03M
	PETAH-1	83.37	**85.43**	**91.22**	65.72	71.17	84.38	80.21	0.06M
	PETAH-2	**83.57**	84.99	91.20	**66.87**	71.10	84.98	**80.43**	0.09M

Table 8. External regularisation for EF L7 full fine-tuning and PETAH-2. We use RandAugment (RA) and Dropout (DO) with $p = 0.1$. Color coding is relative to the baseline without regularization.

Reg.		Full FT						PETAH-2					
RA	**DO**	**CUB**	**Cars**	**Pets**	**Aircr.**	**DTD**	**Mean**	**CUB**	**Cars**	**Pets**	**Aircr.**	**DTD**	**Mean**
✗	✗	89.93	90.12	94.34	72.11	76.31	84.56	89.07	91.20	94.19	75.96	**77.32**	85.55
✓	✗	90.18	91.84	**94.52**	77.59	77.29	86.28	89.23	92.31	94.39	**80.50**	76.12	86.51
✗	✓	90.08	90.44	94.22	72.76	76.60	84.82	88.87	91.16	**94.58**	76.18	76.92	85.54
✓	✓	**90.39**	**91.85**	94.33	**78.13**	**77.71**	**86.48**	**89.40**	**92.80**	94.52	80.35	76.60	**86.73**

Table 9. Fine-Grained Classification and PEFT for sparse backbones trained with 90% sparsity.

	Type	**CUB**	**Cars**	**Pets**	**Aircraft**	**DTD**	**Food**	**Mean**	**#Params**
EF L7-0.9	Linear-probing	86.65	71.12	91.88	53.24	72.93	83.92	76.62	0
	ATTN FT	86.41	86.25	92.70	65.97	73.69	87.32	82.05	1.57M
	Full FT	85.60	88.65	92.57	71.14	73.87	**90.56**	83.73	7.97M
	LoRA ATTN $r = 8$	85.16	88.14	92.65	68.04	72.34	86.62	82.20	0.26M
	PETAH-1	87.48	88.93	93.74	72.50	74.61	89.37	84.44	0.35M
	PETAH-2	**87.82**	**89.64**	**93.89**	**73.51**	75.12	89.47	**84.91**	0.45M
ViT-B-0.9	Linear-probing	87.73	70.84	92.50	50.11	75.05	87.67	77.32	0
	ATTN FT	88.30	89.36	93.61	69.25	76.06	90.44	**84.51**	2.83M
	Full FT	87.30	88.84	93.19	68.13	76.60	89.86	83.99	8.10M
	LoRA ATTN $r = 8$	88.67	88.49	93.77	69.48	76.17	90.12	84.45	0.44M
	Consolidator	88.06	87.09	93.69	66.98	**76.77**	89.51	83.68	0.31M
	SSF	88.07	85.43	93.74	65.34	75.25	87.93	82.63	0.21M