Adapters Strike Back

Supplementary Material

In this appendix, we provide further details and results, which could not be included in the main paper due to space limitations.

A. Why did adapters underperform for ViTs?

First, we want to shed more light on why adapters do not rank well in the literature for parameter-efficient transfer learning for vision tasks. By comparison of numbers reported for adapters on VTAB in the publications referenced in Tab. 4 of the main paper, we found that they essentially stem from only two sources.

The first source is VPT [30], where results for an adapter with a reduction factor of 256, amongst other configurations, are reported. For a ViT-B/16 with a hidden dimension of $d=768$, this is equal to an adapter with rank $r=3$. Despite citing Pfeiffer et al. [47], who suggest a Post-Adapter position, the actual implementation in the code base[3] equals an Intermediate-Adapter that performs worse on VTAB (see Sec. 3.3 of the main paper). The initialization used for the adapter parameters most resembles a LoRA initialization but sets the adapter parameters to zero everywhere. Therefore, there is no randomization in the initialization of the adapter parameters, and different seeds only affect the initialization of the classifier. Additionally, the intermediate features in the adapter bottlenecks then become all zero, leading to identical gradients in the up-projections at the start of training, which hinders optimization. As a result, the adapter baseline used by VPT only reaches 60.0% average accuracy on the VTAB test sets. This is a gap of 17.6 percentage points (pp) compared to our Adapter+ with rank $r=8$ (77.6% average accuracy). Even when considering the loss of around 2–3 pp caused by an unsuitable data normalization in the VPT implementation, this is still a very significant gap. The numbers for an adapter with rank $r=3$ from VPT are also reported in [38] as a baseline.

The second source for adapter baseline results is the NOAH pre-print [63]. There, an adapter with rank $r=8$ is used. Its implementation[4] performs the following feature transformation:

$$\boldsymbol{x} \mapsto \text{Adapter}\big(\text{FFN}(\boldsymbol{x})\big) + \boldsymbol{x}\,. \tag{12}$$

This is closest to the Intermediate-Adapter (*cf*. Eq. (10) of the main paper) but misses the skip connection bypassing the adapter and containing $\text{FFN}(\boldsymbol{x})$. Thus, the adapter does not learn a residual function to an identity mapping but instead must learn a more complex mapping to transform its input. Therefore, the adapter becomes harder to train [23], leading to an average accuracy of 73.9% on the VTAB test sets or 3.7 pp behind our Adapter+. For the NOAH adapter results, we see a proliferation to the publications of FacT [31] and SPT [20]. The adapter implementation from NOAH is also used in the code released for Consolidator[5] [19] but their results are produced with rank $r=16$, giving a slightly better average accuracy of 74.3%, or 3.3 pp less than Adapter+.

In summary, the examined baseline implementations differ from the configurations proposed by Houlsby et al. [27] and Pfeiffer et al. [47] and introduce issues that lead to their underperformance. In our paper, we show that adapters are capable of reaching 77.6% average accuracy for rank $r=8$ and 77.9% for our optimized version of Adapter+, uplifting adapters from an easy-to-beat baseline to a state-of-the-art transfer method.

B. Dataset properties

In Tabs. 8 and 9, we show the statistics of each task in VTAB [62] and FGVC [30] with regard to the number of classes and the number of images in the train, validation, and test splits. The tables are largely "borrowed" from [30].

Table 8. **Dataset details for VTAB.**

Group	Task	# Classes	Splits		
			Train	Val	Test
Natural	CIFAR-100 [34]	100	800	200	10 000
	Caltech-101 [14]	102			6 084
	DTD [8]	47			1 880
	Oxford Flowers [44]	102			6 149
	Pets [46]	37			3 669
	SVHN [43]	10			26 032
	Sun397 [59]	397			21 750
Specialized	Patch Camelyon [57]	2	800	200	32 768
	EuroSAT [24]	10			5 400
	RESISC45 [7]	45			6 300
	Diabetic Retinopathy [13]	5			42 670
Structured	CLEVR-Count [32]	8	800	200	15 000
	CLEVR-Distance [32]	6			15 000
	DMLab [2]	6			22 735
	KITTI-Distance [17]	4			711
	dSprites-Location [41]	16			73 728
	dSprites-Orientation [41]	16			73 728
	smallNORB-Azimuth [35]	18			12 150
	smallNORB-Elevation [35]	9			12 150

[3] https://github.com/KMnP/vpt

[4] https://github.com/ZhangYuanhan-AI/NOAH

[5] https://github.com/THU-MIG/Consolidator

Table 9. **Dataset details for FGVC.** For datasets marked with *, we follow [30] to randomly sample train and validation splits because validation sets are not available from the original datasets.

Dataset	# Classes	Splits		
		Train	Val	Test
CUB-200-2011* [58]	200	5 394	600	5 794
NABirds* [26]	555	21 536	2 393	6 084
Oxford Flowers [44]	102	1 020	1 020	6 149
Stanford Dogs* [33]	120	10 800	1 200	8 580
Stanford Cars* [16]	196	7 329	815	8 041

C. More experimental settings

For all experiments conducted with our implementation, we average the results over three seeds. This includes the (re-)evaluations of LoRA and VPT. We built our implementation on PyTorch [65], PyTorch Lightning,[6] and timm.[7] We run experiments with bfloat16 mixed precision on a NVIDIA RTX A6000 GPU.

For our experiments in the main paper, we report results for a fixed adapter rank r as well as ranks optimized per task. For the per-task optimization of Adapter+, we use a hyper-parameter sweep over the set of ranks $r \in \{1, 2, 4, 8, 16, 32\}$. We evaluate on the validation sets of VTAB and FGVC and choose the per-task ranks from the specified range(s) to steer the number of average parameters. The ranks we used to produce the results on the VTAB and FGVC test sets (see Tabs. 4 and 5 in the main paper) are shown in detail in Tab. 10 and Tab. 11, respectively.

D. Calculation of no. of trainable parameters

Suppose we have a ViT with a hidden dimension d, N transformer layers, and adapters with rank r. The total number of learnable parameters for Adapter$_{\text{base}}$ modules (*cf*. Eq. (4) of the main paper) attached to the FFN of every transformer layer then amounts to $N(2dr + r + d)$. Including layer normalization in the adapter modules amounts to $N2d$ additional parameters. The addition of learned, layer-wise scaling amounts to N extra parameters and choosing learned, channel-wise scaling instead adds Nd extra parameters. Adapter+ (see Sec. 4.3 of the main paper) thus amounts to $N(2dr + 2d + r)$ total parameters. Additionally, for a task with c classes, we add a classifier with $dc + c$ learnable parameters.

E. Vision transformer pre-training

As we add only very few parameters to an otherwise frozen backbone, the generalization capability of the feature representations produced by the backbone is important. For ViTs, there are a number of off-the-shelf models available with differences in their training procedures. Here, we examine three different pre-trainings as examples: *(1) Original*: The ViT-B/16 weights used in the main paper, pre-trained with supervision on ImageNet-21k [53] following the training procedure of the original ViT publication [12],[8] *(2) ImageNet-1k*: the same ViT weights further fine-tuned on ImageNet-1k [53],[9] and *(3) AugReg*: weights from a pre-training with stronger data augmentation in the form of Mixup [67] and RandAugment [64] following [66].[10]

In Tab. 12, we summarize our results for Adapter+ with rank $r = 8$ evaluated on the VTAB validation sets. We notice that additional fine-tuning on ImageNet-1k gives a slight edge (83.4% average accuracy over 83.0% for second best) in adaption for tasks that contain natural images. However, the fine-tuning is detrimental for the Specialized and Structured group. Not fine-tuning on ImageNet-1k is beneficial for the Structured group with a large increase of 3.7 pp. The AugReg training setting improves the transfer to the Specialized group but is worse than the other settings for natural images. Overall, the original supervised training on ImageNet-21k generalizes best across all tasks in VTAB with an average accuracy of 76.5%, 0.3 pp better than AugReg training and 1.2 pp better than ImageNet-1k fine-tuning.

Table 10. **Adapter rank r for each VTAB task** for optimized versions of Adapter+ with different ranges of permitted ranks.

		Natural							Specialized				Structured							
	# Param (M)	CIFAR-100 [34]	Caltech-101 [14]	DTD [8]	Flowers [44]	Pets [46]	SVHN [43]	Sun397 [59]	Camelyon [57]	EuroSAT [24]	RESISC45 [7]	Retinopathy [13]	CLEVR-Count [32]	CLEVR-Dist. [32]	DMLab [2]	KITTI-Dist. [17]	dSpr-Loc. [41]	dSpr-Ori. [41]	sNORB-Azi. [35]	sNORB-Ele. [35]
$r \in [1..4]$	0.11	1	4	2	1	4	4	1	4	2	4	2	4	2	4	4	4	4	4	4
$r \in [1..8]$	0.16	1	4	2	1	8	8	1	8	2	8	8	4	8	8	8	8	4	8	8
$r \in [1..32]$	0.27	1	4	2	1	8	16	1	16	2	32	32	4	8	8	8	32	4	32	8

Table 11. **Adapter rank r for each FGVC dataset** for optimized versions of Adapter+ with different ranges of permitted ranks.

	# Param (M)	CUB-200 [58]	NABirds [26]	Oxford Flowers [44]	Stanford Dogs [33]	Stanford Cars [16]
$r \in [1..32]$	0.34	2	2	1	1	32

[6] https://lightning.ai/pytorch-lightning

[7] https://github.com/huggingface/pytorch-image-models

[8] https://storage.googleapis.com/vit_models/imagenet21k/ViT-B_16.npz

[9] https://storage.googleapis.com/vit_models/imagenet21k+imagenet2012/ViT-B_16-224.npz

[10] https://storage.googleapis.com/vit_models/augreg/B_16-i21k-300ep-lr_0.001-aug_medium1-wd_0.1-do_0.0-sd_0.0.npz

Table 12. **Influence of ViT pre-training.** We use Adapter+ with rank $r=8$ for the evaluation and report the average accuracy in % for each subgroup and across all groups on the VTAB *val sets*.

Pre-training	Natural	Specialized	Structured	Average
ImageNet-1k	**83.4**	86.5	56.0	75.3
AugReg	81.6	**87.2**	**59.7**	76.2
Original	83.0	86.8	**59.7**	**76.5**

Table 13. **Adapter position with DINO backbone.** We report average accuracy in % ($\pm$ std. dev.) on the VTAB *val sets* for different adapter positions. $\text{Adapter}_{\text{base}}$ with Houlsby initialization and rank $r=8$ is used in all experiments.

Position	Natural	Specialized	Structured	Average
Pre	76.8 $\pm$ 0.4	86.2 $\pm$ 0.6	53.6 $\pm$ 0.7	72.2 $\pm$ 0.3
Intermediate	76.8 $\pm$ 0.4	85.8 $\pm$ 0.8	52.6 $\pm$ 0.9	71.8 $\pm$ 0.4
Parallel	76.7 $\pm$ 0.3	**86.8** $\pm$ 0.4	54.1 $\pm$ 0.7	72.5 $\pm$ 0.3
Post	**76.9** $\pm$ 0.2	86.3 $\pm$ 0.5	**55.3** $\pm$ 0.7	**72.8** $\pm$ 0.3

Table 14. **Comparison of Adapter+ with adapter configurations from previous work with DINO backbone.** We report the average accuracy in % ($\pm$ std. dev.) of each subgroup and across all groups on the VTAB *val sets*.

Configuration	# Param	Natural	Specialized	Structured	Average
Houlsby [27], $r=8$	0.39	**77.4** $\pm$ 0.4	**86.5** $\pm$ 0.7	52.9 $\pm$ 0.8	72.3 $\pm$ 0.4
Houlsby [27], $r=4$	0.24	77.2 $\pm$ 0.5	86.2 $\pm$ 0.5	53.2 $\pm$ 0.8	72.2 $\pm$ 0.3
Pfeiffer [47]	0.21	76.8 $\pm$ 0.4	86.2 $\pm$ 0.3	54.4 $\pm$ 1.0	72.5 $\pm$ 0.4
AdaptFormer [6]	**0.19**	76.5 $\pm$ 0.4	85.8 $\pm$ 0.4	53.0 $\pm$ 0.5	71.8 $\pm$ 0.3
Adapter+	0.20	76.7 $\pm$ 0.3	86.4 $\pm$ 0.5	**55.4** $\pm$ 0.8	**72.8** $\pm$ 0.3

F. Generality of the conclusions

Using DINO [5] as an example of a ViT trained with self-supervision, we show in Tab. 13 that the orders of best-to-worst adapter position is consistent with that of a supervised backbone in terms of average accuracy, albeit with a higher standard deviation. The ranking also stays the same for the comparison of Adapter+ with adapter configurations from previous work as presented in Tab. 14. This shows that our conclusions generalize beyond backbones with supervised pre-training to backbones based on self-supervised pre-training.

References

[64] Ekin Dogus Cubuk, Barret Zoph, Jonathon Shlens, and Quoc Le. RandAugment: Practical automated data augmentation with a reduced search space. In *NeurIPS*2020*.

[65] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Z. Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In *NeurIPS*2019*, pages 8024–8035.

[66] Andreas Steiner, Alexander Kolesnikov, Xiaohua Zhai, Ross Wightman, Jakob Uszkoreit, and Lucas Beyer. How to train your ViT? Data, augmentation, and regularization in vision transformers. *Trans. Mach. Learn. Res.*, 2022.

[67] Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *ICLR*, 2018.