Supplementary Material:
Time-, Memory- and Parameter-Efficient Visual Adaptation

Otniel-Bogdan Mercea[1, 2*] Alexey Gritsenko[1] Cordelia Schmid[1] Anurag Arnab[1†]
[1]Google [2]University of Tübingen

In this appendix, we provide further experiments on large-scale image classification (Sec. A), futher details and ablation studies of our baselines (Sec. B), detail our hyper-parameters (Sec. D), and finally, include a more extensive comparison to prior work on the VTAB benchmark [24] (Sec. C) which we could not do in the main paper due to space constraints.

A. Further experiments on large scale image-classification

Table A1 provides detailed results for iNaturalist 2021/2018, Places365 and ImageNet. We also provide inference metrics, such as memory utilization and speed during inference. Similar to the conclusions from the main paper on iNaturalist2018, our proposed LoSA is Pareto-optimal on these datasets, as no other method is both more accurate and more efficient (across multiple efficiency metrics). Figures A1 and A2 visualise our results on iNaturalist 2021 and Places365 respectively.

As mentioned in the main paper, we have also included results for ImageNet in Tab. A1. However, we do not consider ImageNet to be a good dataset for evaluating the performance of adaptation methods, as the performance is saturated, and all methods achieve similar accuracy. The dataset is likely saturated because it is too similar to the pretraining dataset [6, 19, 20, 25]. The lowest accuracy achieved is 88.2 for LST [21], and the highest accuracy is 89.0 for full-finetuning and our method.

Finally, we observe that for inference, the memory consumption and the speed is very similar across all methods. As described in the main paper, this happens as the efficiency metrics during inference are almost entirely influenced by the size of the backbone and not by the size of the adaptors, which is relatively very small. Therefore, the methods that add additional components (*e.g.* LoSA , LST [21], LoRA [8]) behave similarly as the other baselines that do not add any additional components (*e.g.* full finetuning, BitFit [23]) with respect to the efficiency metrics during inference.

B. Further details and ablation studies about baselines

In this section, provide additional details about the baselines used in our main paper. In addition, we have released our code at: https://github.com/google-research/scenic

LoRA [8] We implemented LoRA according to the original paper [8] and the official repository. In the original LoRA paper, the authors only adapted the query- and key-projections within the transformer block. However, as these experiments were conducted for natural language tasks, we performed further ablations of LoRA in Tab. A2 and A3.

Based on our experiments, we found that adapting all linear projections in the transformer block (*i.e.* query-, key-, value- and output-projections, along with the MLP block) performed the best (Tab. A2). Furthermore, we found using a LoRA rank of $r = 32$ to provide a good trade-off between accuracy and efficiency (Tab. A3). As a result, we used these settings in all of our experiments in the main paper.

Finally, we note that during inference, it is possible to absorb the learned LoRA parameters back into the original weights of the transformer block [8]. This means that the inference speed and GFLOPs remains unchanged compared to the backbone. However, we did not implement this in our experiments.

BitFit [23] For BitFit, we implement it so that it can train every single bias in the whole network. This is in line with the original paper [23] which showed that using all the biases in the network provides the best accuracy.

Prompt tuning [9] We implemented visual prompt-tuning, according to the authors' implementation and paper [9]. We have used the "Deep" prompt-tuning, which means that we have learnable tokens at multiple layers in the architecture, instead of only at the first layer as in "Shallow" prompt-tuning.

*Work done during an internship at Google.
†Correspondence to aarnab@google.com.

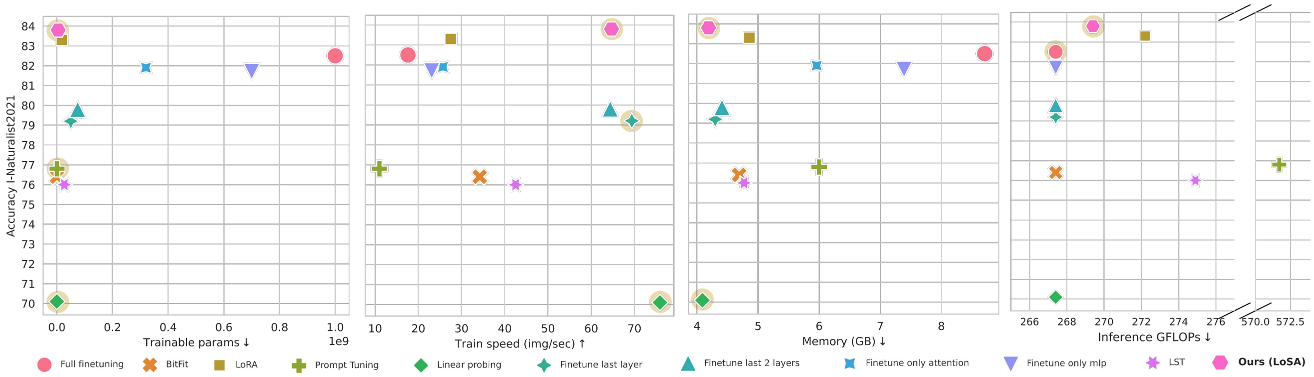

Figure A1. Comparison of trade-offs of accuracy with respect to learned parameters, training memory, inference GFLOPs and training speed. Our approach, LoSA, is consistently on the Pareto frontier (denoted by shaded yellow circles), as there is no method that is both more accurate and more efficient than it, across multiple efficiency metrics. Results are on the iNaturalist2021 dataset, using a ViT-g backbone with 1 billion frozen parameters.

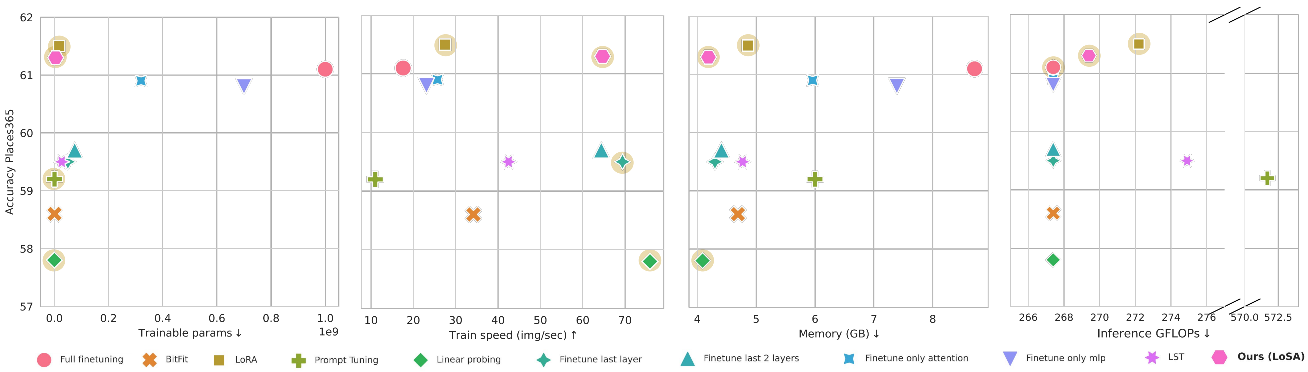

Figure A2. Comparison of trade-offs of accuracy with respect to learned parameters, training memory, inference GFLOPs and training speed. Our approach, LoSA, is consistently on the Pareto frontier (denoted by shaded yellow circles), as there is no method that is both more accurate and more efficient than it, across multiple efficiency metrics. Results are on the Places365 dataset, using a ViT-g backbone with 1 billion frozen parameters.

We found that prompt-tuning is very sensitive to the number of prompts inserted at each layer [8, 9, 13]. And in some cases, can even perform worse than linear probing. Therefore, in our experiments, we first did an ablation study to determine a configuration that achieved good accuracy-efficiency trade-offs, as shown in Tab. A4, and used it for future experiments. Concretely, we used 16 learnable prompts per layer, for the first 24 layers of our ViT-g backbone.

LST [21] Our implementation is based on the public code and paper [21]. The paper uses a transformer model as the parallel network. Following the original authors, we used a linear projection to reduce the dimensionality of the activations from the backbone, to the hidden dimension, d, of the parallel network. Following the authors, we used $d = 48$.

ST-Adapter [17] We have followed the public implementation. and paper [17]. In the paper, the authors provide multiple ways of inserting the ST-Adapter into the backbone architecture. We used the variant which achieves the best accuracy, namely inserting an ST-Adapter module before and after the Multihead Self-Attention (MHSA) in each transformer block. Following the paper, our convolutional kernel size is 3x3x3, and the hidden dimension of the ST-Adapter is 768.

C. VTAB

Table A5 compares to additional methods on the VTAB-1K benchmark (in the main paper, we presented fewer methods due to space constraints). Our LoSA method still achieves superior accuracy-efficiency trade-offs compared to prior work.

D. Implementation details

In this section we provide more details for the hyperparameters that we have used for our methods on the large scale datasets (image and video domains). Table A6 shows that we use the same hyperparameters across different image datasets, and achieve strong results across all of them. Table A7 presents our hyperparameters for video classification.

Our training hyperparameters for image classification are based on those from [19, 25], whilst our training hy-

Method	Update ↓ Param (M)	Memory Usage (GB)		Speed img/sec		GFlops ↓	Acc ↑ Places365	Acc ↑ iNaturalist2018	Acc ↑ iNaturalist2021	Acc ↑ ImageNet
		Train ↓	Inference ↓	Train ↑	Inference↑					
Full fine-tuning	1000	8.71	4.01	17.6	77.3	267.4	61.1	79.8	82.5	89.0
Linear probing	0	4.09	4.01	75.9	77.6	267.4	57.8	73.3	70.1	88.4
LST [21]	27	4.77	4.11	42.5	66.4	274.9	59.5	75.0	76.0	88.2
Finetune last layer	50	4.30	4.01	69.4	77.5	267.4	59.5	77.4	79.2	88.8
Prompt Tuning [9]	0.6	6.00	4.04	11.0	27.9	571.8	59.2	77.5	76.8	88.3
Finetune last 2 layers	75	4.41	4.01	64.4	77.6	267.4	59.7	78.1	79.8	88.8
Bitfit [23]	0.7	4.69	4.01	34.2	77.5	267.4	58.6	78.6	76.4	88.4
Finetune only mlp [22]	700	7.39	4.01	23.1	77.7	267.4	60.8	79.3	81.7	88.9
Finetune only attention [22]	320	5.96	4.01	25.7	77.4	267.4	60.9	79.7	81.9	88.8
LoRA [8]	18	4.86	4.08	27.5	64.2	272.2	61.5	80.9	83.3	88.6
Ours (LoSA)	4.8	4.19	4.07	64.7	73.6	269.4	61.3	81.3	83.8	89.0

Table A1. Results on iNaturalist2018/2021, Places365 and ImageNet. We used a Vit-g backbone with 1 billion paramaeters for all experiments. The number of trainable parameters does not contain the classifier weights, which means that the number of updated parameters is therefore constant across all datasets. The speed, memory usage and GFLOPs are barely affected by the number of classes, and we report these metrics for iNaturalist 2018. We used data from this table for the Figures 1 and 4 of the main paper, and Fig. A1 and A2 of this appendix. Note that for LoRA [8], it is possible to absorb the learned weights into the frozen weights after training, to ensure that the inference time and FLOPs does not change at all compared to full finetuning which does not add any parameters to the model [8]. However, we have not implemented this. Note that although we average training- and inference-time over 50 batches, there is still some random variation in the timing.

Q	K	V	Out	MLP	Params (M)	GFLOPs	Time (img/sec)	Accuracy
✓					3.7	268.4	30.5	77.7
✓	✓				7.4	269.4	28.9	77.7
✓	✓	✓			11.1	270.3	27.3	79.7
✓	✓	✓	✓		14.8	271.2	26.5	79.9
✓	✓	✓	✓	✓	18.5	272.2	27.5	80.9

Table A2. Ablation on which components of the transformer block to apply LoRA to, on the iNaturalist2018 dataset, using a Vit-g backbone. Our choice of using LoRA on all components provides the best accuracy.

Rank, r	Params (M)	GFLOPs	Time (img/sec)	Accuracy
1	0.58	267.7	26.8	79.2
4	2.31	268.1	27.7	79.0
8	4.62	268.7	27.8	79.9
16	9.24	269.8	27.7	80.6
32	18.47	272.2	27.5	80.9
64	36.95	276.8	27.3	81.1
128	73.89	286.1	26.7	80.5
256	147.79	304.8	25.0	79.8

Table A3. Ablating the LoRA rank on the iNaturalist2018 dataset, using Vit-g backbone. We consider our choice of using $r = 32$ a good trade-off.

perparameters for video classification are based on those from [1]. Overall, we did not change hyperparametres of our model (such as the rank r of LoSA) between images and video.

We used synchronous SGD with distributed data-parallel training. For images, we used a local batch size of 16. For video, our local batch size was 1 for all experiments in the main paper.

Number of prompts, P	Number of initial layers, L	Params (10^3)	GFLOPs	Accuracy
0	0	0	267.4	73.3
1	1	1	268.5	73.4
1	4	6	271.6	72.9
1	8	11	275.3	72.7
1	16	23	281.4	72.8
1	24	34	285.9	72.4
4	1	6	271.7	72.5
4	4	23	284.0	72.6
4	8	45	298.9	70.9
4	16	90	323.6	72.4
4	24	135	341.6	75.4
8	1	11	276.0	72.7
8	4	45	300.6	71.4
8	8	90	330.5	70.5
8	16	180	380.5	73.8
8	24	270	417.0	77.1
16	1	23	284.6	72.3
16	4	90	334.0	70.6
16	8	180	394.5	70.8
16	16	361	496.4	75.1
16	24	541	571.8	77.5
24	1	34	293.3	72.4
24	4	135	376.6	70.1
24	8	270	459.2	73.0
24	16	541	615.0	74.7
24	24	811	731.6	78.4

Table A4. Ablating prompt tuning on the iNaturalist2018 dataset with a Vit-g backbone. Prompt tuning is sensitive to the number of learned prompts, P, and the number of initial layers, L, which these prompts are added to. (Note that prompts are not added only to the input, but the initial L layers in [9]). $P = 0, L = 0$, corresponds to a linear probing baseline. Based on these results, we used $P = 16$, $L = 24$.

When we report the memory consumption for training or inference (Fig. 1 and 4, Tabs. 2, 3, 4, 5 of the main paper, and Fig. A1 and A2 and Tab. A1 of the appendix, we report the memory usage with a local batch size of 1. The reason for that is that it removes the effect of the batch size (which is a training hyperparameter), and it also has a clear intepre-

		Natural							Specialized				Structured											
	Param ↓ (10^6)	Cifar100	Caltech101	DTD	Flower102	Pets	SVHN	Sun397	Camelyon	EuroSAT	Resisc45	Retinopathy	Clevr-Count	Clevr-Dist	DMLab	KITTI-Dist	dSpr-Loc	dSpr-Ori	sNORB-Azim	sNORB-Ele	Avg Natural	Avg Specialized	Avg Structured	Average
Traditional Finetuning																								
Full[9, 11]	85.8	68.9	87.7	64.3	97.2	86.9	87.4	38.8	79.7	95.7	84.2	73.9	56.3	58.6	41.7	65.5	57.5	46.7	25.7	29.1	75.9	83.4	47.6	68.9
Linear[9, 11]	0	64.4	85.0	63.2	97.0	86.3	36.6	51.0	78.5	87.5	68.5	74.0	34.3	30.6	33.2	55.4	12.5	20.0	9.6	19.2	69.1	77.1	26.9	57.6
Efficient adaptation methods																								
BitFit[12, 23]	0.10	72.8	87.0	59.2	97.5	85.3	59.9	51.4	78.7	91.6	72.9	69.8	61.5	55.6	32.4	55.9	66.6	40.0	15.7	25.1	73.3	78.3	44.1	65.2
VPT-Shal.[9, 12]	0.06	77.7	86.9	62.6	97.5	87.3	74.5	51.2	78.2	92.0	75.6	72.9	50.5	58.6	40.5	67.1	68.7	36.1	20.2	34.1	76.8	79.7	47.0	67.8
VPT-Deep[9, 11]	0.53	78.8	90.8	65.8	98.0	88.3	78.1	49.6	81.8	96.1	83.4	68.4	68.5	60.0	46.5	72.8	73.6	47.9	32.9	37.8	78.5	82.4	55.0	72.0
RS-Bypass. [10]	0.42	64.5	88.8	73.2	99.4	90.6	63.5	57.2	85.5	95.2	82.4	75.2	70.4	61.0	40.2	66.8	79.2	52.6	26.0	49.3	76.7	84.6	55.7	72.3
Express [4]	0.2*	78.0	89.6	68.8	98.7	88.9	81.9	51.9	84.8	96.2	80.9	74.2	66.5	60.4	46.5	77.6	78.0	49.5	26.1	35.3	79.7	84.0	55.0	72.9
TOAST [18]	14.0	82.1	90.5	70.5	98.7	89.7	71.9	53.3	84.3	95.5	85.5	74.2	75.4	60.8	44.7	77.5	73.9	47.5	24.5	33.7	79.5	84.9	54.8	73.1
Adapter [7, 11]	0.16	69.2	90.1	68.0	98.8	89.9	82.8	54.3	84.0	94.9	81.9	75.5	80.9	65.3	48.6	78.3	74.8	48.5	29.9	41.6	79.0	84.1	58.5	73.9
LoRA[8, 11]	0.29	67.1	91.4	69.4	98.8	90.4	85.3	54.0	84.9	95.3	84.4	73.6	82.9	**69.2**	49.8	78.5	75.7	47.1	31.0	44.0	79.5	84.6	59.8	74.5
AdaptFormer[3, 11]	0.16	70.8	91.2	70.5	99.1	90.9	86.6	54.8	83.0	95.8	84.4	76.3	81.9	64.3	49.3	80.3	76.3	45.7	31.7	41.1	80.6	84.9	58.8	74.7
NOAH [11, 26]	0.36	69.6	92.7	70.2	99.1	90.4	86.1	53.7	84.4	95.4	83.9	75.8	82.8	68.9	49.9	81.7	81.8	48.3	32.8	44.2	80.3	84.9	61.3	75.5
FacT-TK [12]	0.06	70.6	90.6	70.8	99.1	90.7	88.6	54.1	84.8	96.2	84.5	75.7	82.6	68.2	49.8	80.7	80.8	47.4	33.2	43.0	80.6	85.3	60.7	75.6
SSF [14]	0.24	69.0	92.6	75.1	99.4	91.8	90.2	52.9	**87.4**	95.9	87.4	75.5	75.9	62.3	**53.3**	80.6	77.3	54.9	29.5	37.9	81.6	86.6	59.0	75.7
$\text{Convpass}_{\text{attn}}$ [11]	0.16	71.8	90.7	72.0	99.1	91.0	89.9	54.2	85.2	95.6	83.4	74.8	79.9	67.0	50.3	79.9	84.3	53.2	34.8	43.0	81.2	84.8	61.6	75.8
RepAdapter [16]	0.22	72.4	91.6	71.0	99.2	91.4	90.7	55.1	85.3	95.9	84.6	75.9	82.3	68.0	50.4	79.9	80.4	49.2	38.6	41.0	81.6	85.4	61.2	76.1
RS [10]	0.55	75.2	92.7	71.9	99.3	**91.9**	86.7	**58.5**	86.7	95.6	85.0	74.6	80.2	63.6	50.6	80.2	85.4	55.7	31.9	42.0	82.3	85.5	61.2	76.3
Convpass [11]	0.33	72.3	91.2	72.2	99.2	90.9	**91.3**	54.9	84.2	96.1	85.3	75.6	82.3	67.9	51.3	80.0	85.9	53.1	36.4	44.4	81.7	85.3	62.7	76.6
HST [15]	0.78	76.7	**94.1**	74.8	99.6	91.1	91.2	52.3	87.1	96.3	**88.6**	76.5	**85.4**	63.7	**52.9**	81.7	87.2	56.8	35.8	**52.1**	**82.8**	**87.1**	64.5	78.1
LoSA , $r = 16$	0.19	82.5	92.8	76.1	**99.7**	90.5	82.0	55.8	86.6	**97.1**	87.0	**76.7**	81.5	62.3	48.6	82.1	**94.2**	**61.7**	47.9	45.6	**82.8**	86.9	**65.5**	**78.4**
LoSA , $r = 8$	0.10	82.2	92.7	**76.7**	**99.7**	90.7	81.0	55.4	86.9	**97.1**	87.4	76.5	79.9	61.8	48.6	82.4	92.3	61.1	**48.7**	47.3	82.6	87.0	65.3	78.3
LoSA , $r = 4$	**0.05**	**82.7**	93.0	76.2	**99.7**	89.8	80.0	56.1	86.3	96.7	86.7	76.3	78.8	61.4	48.0	**82.6**	91.7	58.4	46.9	47.6	82.5	86.5	64.4	77.8

Table A5. Comparison to state-of-the-art parameter-efficient finetuning methods on VTAB-1K [24]. Following standard practice, the final "Average" is the average of three preceding groupwise averages. Parameters denotes the number of learnable parameters excluding the final classification layer, as the number of parameters in this final layer depend on the number of classes, which varies between 2 and 397. Each variant of our model, which we obtain by varying the rank r of our Low-rank Mixer Block, achieves better accuracy-parameter trade-offs than previous approaches, when using the same ViT-B backbone. **Best results** are bolded, and second-best underlined. * denotes that we estimated the number of learnable parameters in Express [4] based on the hyperparameters presented in the paper, as the authors did not explicitly state the number of learned parameters in their paper.

Hyperparameter	iNaturalist2018	iNaturalist2021	Places365	ImageNet
Optimiser	Momentum, $\lambda = 0.9$			
Batch size	512			
Learning rate scheduler	cosine			
Linear warmup steps	500			
Base learning rate	0.05			
Number of training steps	20 000			
Rank, r	64			
Input resolution	224			

Table A6. LoSA hyperparameters for large-scale image classification datasets.

Hyperparameter	Kinetics 400
Optimiser	momentum, $\lambda = 0.9$
Batch size	64
Learning rate scheduler	cosine
Linear warmup epochs	2.5
Base learning rate	0.04
Epochs	30
Rank, r	64
Input resolution	224

Table A7. LoSA hyperparameters for video classification task.

tation: it represents the minimum possible memory that we require during data-parallel training.

When reporting the training and inference speeds, we average the runtime over 50 batches.

We implemented our method and baselines using the Scenic library [5] and JAX [2].

References

[1] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lučić, and Cordelia Schmid. ViViT: A Video Vision Transformer. In *ICCV*, 2021. 3

[2] James Bradbury, Roy Frostig, Peter Hawkins,

Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. 4

[3] Shoufa Chen, Chongjian Ge, Zhan Tong, Jiangliu Wang, Yibing Song, Jue Wang, and Ping Luo. Adaptformer: Adapting vision transformers for scalable visual recognition. In *NeurIPS*, 2022. 4

[4] Rajshekhar Das, Yonatan Dukler, Avinash Ravichandran, and Ashwin Swaminathan. Learning expressive prompting with residuals for vision transformers. In *CVPR*, 2023. 4

[5] Mostafa Dehghani, Alexey Gritsenko, Anurag Arnab, Matthias Minderer, and Yi Tay. Scenic: A JAX library for computer vision research and beyond. *arXiv preprint arXiv:2110.11403*, 2021. 4

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *CVPR*, 2009. 1

[7] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *ICML*, 2019. 4

[8] 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. In *ICLR*, 2022. 1, 2, 3, 4

[9] Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge Belongie, Bharath Hariharan, and Ser-Nam Lim. Visual prompt tuning. In *ECCV*, 2022. 1, 2, 3, 4

[10] Zeyinzi Jiang, Chaojie Mao, Ziyuan Huang, Ao Ma, Yiliang Lv, Yujun Shen, Deli Zhao, and Jingren Zhou. Res-tuning: A flexible and efficient tuning paradigm via unbinding tuner from backbone. In *NeurIPS*, 2023. 4

[11] Shibo Jie and Zhi-Hong Deng. Convolutional bypasses are better vision transformer adapters. In *arXiv preprint arXiv:2207.07039*, 2022. 4

[12] Shibo Jie and Zhi-Hong Deng. Fact: Factor-tuning for lightweight adaptation on vision transformer. In *AAAI*, 2023. 4

[13] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *ACL*, 2021. 2

[14] Dongze Lian, Daquan Zhou, Jiashi Feng, and Xinchao Wang. Scaling & shifting your features: A new baseline for efficient model tuning. In *NeurIPS*, 2022. 4

[15] Weifeng Lin, Ziheng Wu, Jiayu Chen, Wentao Yang, Mingxin Huang, Jun Huang, and Lianwen Jin. Hierarchical side-tuning for vision transformers. In *arXiv preprint arXiv:2310.05393*, 2023. 4

[16] Gen Luo, Minglang Huang, Yiyi Zhou, Xiaoshuai Sun, Guannan Jiang, Zhiyu Wang, and Rongrong Ji. Towards efficient visual adaption via structural re-parameterization. In *arXiv preprint arXiv:2302.08106*, 2023. 4

[17] Junting Pan, Ziyi Lin, Xiatian Zhu, Jing Shao, and Hongsheng Li. St-adapter: Parameter-efficient image-to-video transfer learning. In *NeurIPS*, 2022. 2

[18] Baifeng Shi, Siyu Gai, Trevor Darrell, and Xin Wang. Toast: Transfer learning via attention steering. In *arXiv preprint arXiv:2305.15542*, 2023. 4

[19] 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. *TMLR*, 2022. 1, 2

[20] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *ICCV*, 2017. 1

[21] Yi-Lin Sung, Jaemin Cho, and Mohit Bansal. Lst: Ladder side-tuning for parameter and memory efficient transfer learning. In *NeurIPS*, 2022. 1, 2, 3

[22] Hugo Touvron, Matthieu Cord, Alaaeldin El-Nouby, Jakob Verbeek, and Hervé Jégou. Three things everyone should know about vision transformers. In *ECCV*, 2022. 3

[23] Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In *arXiv preprint arXiv:2106.10199*, 2021. 1, 3, 4

[24] Xiaohua Zhai, Joan Puigcerver, Alexander Kolesnikov, Pierre Ruyssen, Carlos Riquelme, Mario Lucic, Josip Djolonga, Andre Susano Pinto, Maxim Neumann, Alexey Dosovitskiy, et al. A large-scale study of representation learning with the visual task adaptation benchmark. In *arXiv preprint arXiv:1910.04867*, 2019. 1, 4

[25] Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers. In *CVPR*, 2022. 1, 2

[26] Yuanhan Zhang, Kaiyang Zhou, and Ziwei Liu. Neural prompt search. In *arXiv preprint arXiv:2206.04673*, 2022. 4