Conv-Adapter: Exploring Parameter Efficient Transfer Learning for ConvNets

Supplementary Material

6. Implementation Details

In this section, we provide more implementation details of Conv-Adapter. We first show the details of the datasets we used and the pre-trained models we used. Then we present the details of hyper-parameter used for each method and each dataset in experiments. We implement all ConvNets and Conv-Adapter in PyTorch, and the code will be made available.

6.1. Datasets

The specifications of the all datasets evaluated in experiments are shown in Tab. 6.

Table 6. Specification of all datasets evaluated. We use * to indicated randomly sampled train and validation sets (from original training set) for datasets which do not have validation split.

Dataset	Description	# Class	# Train	# Val	# Test
CUB-200-2011	FGVC	200	5,394*/5,994	600*	5,794
NABirds		700	21,536*/23,929	2,393*	24,633
Stanford Dogs		120	10,800*/12,000	1,200*	8,580
Stanford Cars		196	7,329*/8,144	815*	8,041
CIFAR-100	Natural	100	800/200	200	10,000
Caltech101		102			6,084
DTD		47			1,880
Oxford Flowers102		102			6,149
Oxford Pets		37			3,669
SVHN		10			26,032
Sun397		397			21,750
Patch Camelyon	Specialized	2	800/200	200	32,768
EuroSAT		10			5,400
Resisc45		45			6,300
Retinopathy		5			42,670
Clevr/count	Structured	8	800/200	200	15,000
Clevr/dist		6			15,000
DMLab		6			22,735
KITTI/dist		4			711
dSprites/loc		16			73,728
dSprite/ori		16			73,728
SmallNORB/azi		18			12,150
SmallNORB/ele		9			12,150
FGVCAirCraft	Few-Shot	102	shots × classes	3,333	3,333
Food101		101		20,200	30,300
Oxford Flowers102		102		1,633	2,463
Oxford Pets		37		736	3,669
Stanford Cars		196		1,635	8,041
MS-COCO	Detection	80	117,266	5,000	-
ADE-20k	Segmentation	150	25,574	2,000	-

6.2. Models

We present the details of the pre-trained models used in experiments in Tab. 7, with the checkpoint link.

6.3. Hyper-parameters in Experiments

We provide the hyper-parameters search range and important settings used in experiments in this section. The detailed hyper-parameters used in experiments will be made available as configuration files in code.

6.3.1 Classification on FGVC and VTAB-1k

For classification tasks of FGVC and VTAB-1k, we summarize the hyper-parameter range in Tab. 8. For VTAB-1k, we use the recommended optimal data augmentations in [60], rather than solely Resize and Centre Crop as in [63]. We find the recommended augmentations produces better results for full-tuning. For FGVC, we use RandomResized Crop with a minimum scale of 0.2 and Horizontal Flip [50] as augmentation. For few-shot classifications, we use the same range as in Tab. 8 and same augmentations as for FGVC tasks.

6.4. Dense Prediction Tasks

6.4.1 Object Detection

We compare all four schemes of Conv-Adapter with the fine-tuning baseline. Specifically, we follow a standard 1x training schedule: all models are trained with a batch size of 16 and optimized by AdamW with an initial learning rate of 0.0002 for Faster RCNN and 0.0001 for RetinaNet, which are then dropped by a factor of 10 at the 8-th and 11-th epoch. The shorter side of the input image is resized to 800 while maintaining the original aspect ratio.

6.4.2 Semantic Segmentation

We conduct similar experiments for the segmentation task. For ResNet50 backbones, we train all models for 80k iterations with an random cropping augmentation of 512×512 input resolution. For ConvNeXt models, we use a larger input resolution of 640×640 and train the models for 160k iterations. We apply AdamW optimizer with a polynomial learning rate decay schedule.

7. Extended Analysis

7.1. Model Analysis

In this section, we provide an analysis of the trainable parameters, model latency, and GFLOPs, based on ResNet50 [19] and ConvNext-B [36]. Since Conv-Adapter is applied on each residual block, we first provide a theoretical analysis of the trainable parameters of each adapting scheme

Table 7. Specification of pre-trained models used in experiments.

Backbone	Pre-trained Objective	Pre-trained Dataset	# Param (M)	Feature Dim
ResNet50 [19]	Supervised	ImageNet-1k	23.5	2,048
ResNet50 [19]	Supervised	ImageNet-21k	23.5	2,048
ResNet50 BiT-M [27]	Supervised	ImageNet-21k	23.5	2,048
ConvNext-B [36]	Supervised	ImageNet-1k	87.6	1,024
ConvNext-B [36]	Supervised	ImageNet-21k	87.6	1,024
ConvNext-L [36]	Supervised	ImageNet-21k	196.2	1,536
Swin-B [34]	Supervised	ImageNet-21k	86.7	1,024
Swin-L [34]	Supervised	ImageNet-21k	194.9	1,536
ResNet50 [45]	CLIP	CLIP	38.3	1,024
ResNet50x4 [45]	CLIP	CLIP	87.1	640
ResNet50 [20]	MoCov3	ImageNet-1k	23.5	2,048

Table 8. Hyper-parameter range for grid-search on image classification tasks of FGVC and VTAB-1k.

	All Backbones
Optimizer	AdamW [37]
LR Range	[1e-3, 5e-4, 1e-4, 5e-5, 1e-5]
WD Range	[1e-2, 1e-3, 1e-4, 0]
LR schedule	cosine
Total Epochs	100
Warmup	10

proposed in Tab. 9. Take the bottleneck residual block of ResNet50 as an example, we set the channel size for each convolution in the residual block as C_{in}, C_{mid}, and C_{out} respectively, where C_{in} is usually set to $\frac{C_{in}}{4}$. We assume the spatial size of the feature maps do not change at each residual block.

We also provide the measurement of training/testing latency, memory cost, and GFLOPs for all the tasks evaluated in this paper, as shown in Tab. 10. For image classification, we average the inference speed over a batch of 64. Although Conv-Adapter has increased testing latency because of the inference includes forwarding on both backbone and Conv-Adapter, the latency and memory cost of training is not necessarily greater thanks to reduced overhead of gradient computation.

Table 9. Analysis of trainable parameters of the 4 proposed adapting schemes, compared to fine-tuning.

Tuning	Input	Output	Trainable Param.
FT	$C_{in} \times H \times W$	$C_{out} \times H \times W$	$K \times K \times C_{in} \times C_{mid} + C_{in} \times C_{mid} + C_{out} \times C_{mid}$
Conv. Par	$C_{mid} \times H \times W$	$C_{mid} \times H \times W$	$K \times K \times C_{mid} + \frac{C_{mid}}{\gamma} \times C_{mid}$
Conv. Seq.	$C_{mid} \times H \times W$	$C_{mid} \times H \times W$	$K \times K \times C_{mid} + \frac{C_{mid}}{\gamma} \times C_{mid}$
Res. Par.	$C_{in} \times H \times W$	$C_{out} \times H \times W$	$K \times K \times C_{in} + \frac{C_{in}}{\gamma} \times C_{in}$
Res. Seq.	$C_{in} \times H \times W$	$C_{out} \times H \times W$	$K \times K \times C_{in} + \frac{C_{in}}{\gamma} \times C_{in}$

Table 10. Evaluation of model latency, memory, and GFLOPs of 4 proposed variants for ResNet-50 and ConvNext-B on image classification, object detection, and semantic segmentation

Image Classification, Input Res. (224 × 224)						
Backbone	Tuning	Train		Test		GFLOPs
		Latency (ms/img)	Memory (GB)	Latency (ms/img)	Memory (GB)	
ResNet50-BiT	FT	1.40	7.46	0.43	2.81	4.12
	Conv. Par.	1.21	7.35	0.48	2.81	4.34
	Conv. Seq.	1.24	7.62	0.54	2.81	4.34
	Res. Par.	1.81	8.45	0.69	2.85	7.0
	Res. Seq.	1.83	9.78	0.72	2.83	7.47
ConvNeXt-B	FT	4.18	16.96	1.17	2.92	15.36
	Conv. Par.	4.91	13.52	1.70	2.98	17.53
	Conv. Seq.	4.94	14.55	1.70	2.98	17.53
	Res. Par.	4.84	13.50	1.75	2.98	17.53
	Res. Seq.	4.84	14.76	1.72	2.99	17.6
Object Detection (Test only)						
Backbone	Tuning	Input Res.		Latency (ms/img)		GFLOPs
ResNet50	FT	1280 × 800		9.38		84.08
	Conv. Par.			9.37		88.61
	Conv. Seq.			11.00		88.61
	Res. Par.			17.09		142.89
	Res. Seq.			16.30		152.54
ConvNeXt-B	FT	1280 × 800		28.55		313.45
	Conv. Par.			41.00		357.66
	Conv. Seq.			41.03		357.66
	Res. Par.			41.13		357.66
	Res. Seq.			41.22		359.22
Semantic Segmentation (Test only)						
ResNet50	FT	2048 × 1024		17.18		172.19
	Conv. Par.			17.21		181.47
	Conv. Seq.			20.23		181.47
	Res. Par.			29.79		292.63
	Res. Seq.			21.22		312.4
ConvNeXt-B	FT	2048 × 1024		58.67		641.95
	Conv. Par.			83.69		732.48
	Conv. Seq.			83.77		732.48
	Res. Par.			83.07		732.48
	Res. Seq.			84.21		735.69

7.2. More Ablation of Conv-Adapter

7.3. CKA Similarity Analysis

While the accuracy performance is well compared for PET methods, theoretical understandings towards under which circumstances PET works better than Fine-tuning lack discovery yet. In this section, we study how weights of backbones change with Fine-tuning using Centered Kernel Analysis and empirically discover insightful observations.

7.3.1 Similarity Measurement between Filter Weights using CKA

As shown in the experimental results, whether the performance of PET surpasses Fine-tuning varies from datasets and domains. From the perspective of trainable weights, PET replaces the whole backbone with much smaller number of parameters compared with Fine-tuning. With the pre-trained backbone and the fine-tuned backbone, we first compute the similarity between the weights of each convolution filter using Centered Kernel Alignment (CKA). In doing so, the changes of weights brought by Fine-tuning are quantified by similarity distances between filters.

CKA is used to compute the representation similarity between hidden layers of neural networks [28, 47]. By inputting matrices $\mathbf{X} \in \mathbb{R}^{n \times m_1}$, $\mathbf{Y} \in \mathbb{R}^{n \times m_2}$, and their Gram matrices $\mathbf{K} = \mathbf{X}\mathbf{X}^T$ and $\mathbf{L} = \mathbf{Y}\mathbf{Y}^T$, CKA follows:

$$\mathrm{CKA}(\mathbf{K}, \mathbf{L}) = \frac{\mathrm{HSIC}(K, L)}{\sqrt{\mathrm{HSIC}(K, K)\,\mathrm{HSIC}(L, L)}},$$

Where HSIC is the Hilbert-Schmidt independence criterion. Instead of analyzing the representation similarity, we focus on analyzing how filter weights change by Fine-tuning and utilize CKA to compute the similarity between filter weights. For each $k \times k$ convolution layer with c_1 input channels and c_2 output channels, weights from the initial pre-trained backbone is referred as $\mathbf{W}_i$ and weights from the fine-tuned backbone is referred as $\mathbf{W}_f$. The filter weights are reshaped into matrices for CKA computation:

- For $k = 1$, the convolutional filter serves as a linear transformation between channels. When computing CKA similarity, $X = \mathbf{W}_i, \mathbf{W}_i \in \mathbb{R}^{c_1 \times c_2}$ and $Y = \mathbf{W}_f, \mathbf{W}_f \in \mathbb{R}^{c_1 \times c_2}$.
- for $k > 1$, the weight matrix can be viewed as $c_1 \times c_2$ filters and each filter carries a size of $k \times k$ weights. When computing CKA similarity, $X = \mathbf{W}_i, \mathbf{W}_i \in \mathbb{R}^{k^2 \times c_1 c_2}$, $Y = \mathbf{W}_f, \mathbf{W}_f \in \mathbb{R}^{k^2 \times c_1 c_2}$.

For each ConvNet, we compute the average of CKA similarities among all convolutional filters and show the results of ResNet, ConvNext and ResNet-CLIP in Fig. 7. With a relatively low accuracy of Finetuning, the similarity between filter weights may not be well representative due to insufficient optimization. Thus NABirds is removed in the analysis. We also measure the domain difference between datasets with ImageNet1k using Maximum Mean Discrepancy (Details are in the following section). Firstly we observe that with less domain difference between target dataset and pre-trained dataset, the Conv-Adapter achieves closer performance with fully finetuning. Secondly, as shown in Fig. 7, accuracy gain of Conv-Adapter and CKA similarities between filter weights share the same trends over datasets and this phenomenon generalizes over different architectures. *When fully finetuning only leads to small changes on filter weights (larger CKA similarities), Conv-Adapter is more likely to surpass the performance of fully finetuning.*

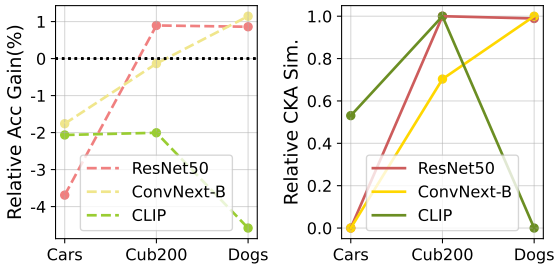

Figure 7. CKA Similarity and Accuracy gap between Conv-Adapter and fully fine-tuning for FGVC datasets.

7.3.2 Domain Difference Quantification using MMD (Maximum Mean Discrepancy)

Maximum Mean Discrepancy (MMD): measures the distance between two data distributions p and q. $\phi()$ refers to a feature extractor (could be a functional intermediate layer):

$$\mathrm{MMD}(p, q) = \|\mathbf{E}_p[\phi(\mathbf{x})] - \mathbf{E}_q[\phi(\mathbf{x})]\|^2_{\mathcal{H}_k}, \quad (3)$$

where $\mathcal{H}_k$ refers to the kernel Hilbert space. We consider the domain difference between ImageNet1k and each dataset from FGVC. Specifically, The features subtracted from pre-trained backbones namely ResNet50 (pre-trained by Imagenet1K, ImageNet21K and CLIP), ConvNext-B (pretrained by ImageNet1K and ImageNet21K). MMD with Gaussian Kernel is computed using features from each backbone and the average MMD over all backbones is used in Fig. 7.

8. Supplementary Results

In this section, we provide some supplementary results to the main paper.

8.1. Detailed Results on VTAB-1k

We provide the per-task results on VTAB-1k on ResNet50 BiT-M [27] and ConvNext-B [36] in Tab. 11 and Tab. 12 respectively.

Table 11. Per-task VTAB-1k results of ImageNet-21k pretrained ResNet50 BiT-M.

Tuning	# Param	Caltech101	Cifar100	DTD	Flowers102	Pets	Sun397	SVHN	Patch Camelyon	EuroSAT	Resisc45	Diabetic Retinopathy	Clevr/count	Clevr/dist	Dmlab	Dsprites/loc	Dsprites/ori	Kitti	Smallnorb/azi	Smallnorb/ele
FT	23.63	84.79±0.46	48.28±0.56	65.32±0.3	97.5±0.05	86.74±0.46	38.14±0.24	84.57±0.91	85.2±0.39	95.46±0.17	84.03±0.15	78.74±0.11	96.77±1.37	58.15±0.3	51.17±0.08	94.39±0.96	69.77±0.68	78.99±0.46	41.79±0.62	42.74±0.19
LP	0.11	84.35±0.51	44.02±0.18	66.49±0.15	98.85±0.03	88.16±0.23	43.24±0.58	46.8±0.06	79.88±0.33	92.53±0.15	78.65±0.24	74.64±0.08	50.43±0.09	33.91±0.19	37.92±0.16	34.23±0.07	33.67±0.09	66.95±0.34	18.27±0.19	27.96±0.09
Bias	0.15	83.75±0.08	41.99±0.4	66.31±0.35	97.84±0.04	87.91±0.45	39.29±0.21	45.34±0.3	79.82±0.19	91.07±0.03	75.77±0.62	74.72±0.04	41.97±0.13	33.27±0.17	37.86±0.03	18.4±0.13	19.43±0.43	67.32±0.26	13.59±0.23	25.55±0.44
VPT	0.15	83.4±0.87	34.92±0.15	59.06±0.13	98.1±0.38	86.14±0.37	43.34±0.22	53.08±0.31	81.06±0.99	91.04±0.09	75.07±0.21	74.25±0.09	49.2±0.43	46.25±0.31	38.64±0.16	41.87±0.93	33.53±2.25	43.84±31.0	20.6±0.53	27.2±0.49
Conv. Par.	0.48	85.26±0.49	48.29±0.07	68.79±0.44	98.28±0.18	86.16±0.03	43.9±0.34	77.55±0.18	84.25±0.59	95.45±0.13	80.67±0.17	76.48±0.21	78.57±1.45	49.17±0.42	46.37±0.73	68.3±6.06	70.55±0.75	78.11±0.52	27.84±0.71	34.69±0.22
Conv. Seq.	0.67	83.43±0.49	48.92±0.38	68.09±0.64	97.89±0.25	85.75±0.35	42.78±0.17	79.11±1.13	84.08±0.55	94.23±0.17	80.78±0.53	76.32±0.09	73.73±2.04	50.61±0.47	46.16±0.2	85.51±3.07	71.7±0.63	75.76±1.57	30.52±0.24	34.03±0.22
Res. Par.	4.61	85.94±0.57	44.2±0.77	67.29±0.67	98.1±0.02	86.57±0.6	40.4±1.93	79.75±0.61	84.07±0.62	94.84±0.31	83.3±0.13	76.59±0.09	84.18±1.87	54.83±1.03	45.42±0.61	95.78±0.23	66.81±0.58	76.98±0.59	30.72±0.62	35.97±0.43
Res. Seq.	7.06	85.4±0.49	45.27±0.76	65.44±0.37	98.18±0.05	86.21±0.17	42.18±0.1	79.53±0.32	84.9±0.37	95.38±0.12	82.43±0.52	76.67±0.15	79.23±1.13	56.54±1.45	48.02±0.58	96.38±0.62	70.41±0.23	72.85±1.21	31.17±1.0	36.05±0.08

Table 12. Per-task VTAB-1k results of ImageNet-21k pre-trained ConvNext-B.

Tuning	# Param	Caltech101	Cifar100	DTD	Flowers102	Pets	Sun397	SVHN	Patch Camelyon	EuroSAT	Resisc45	Diabetic Retinopathy	Clevr/count	Clevr/dist	Dmlab	Dsprites/loc	Dsprites/ori	Kitti	Smallnorb/azi	Smallnorb/ele
FT	87.62	91.97±0.69	69.06±0.42	76.15±0.28	99.55±0.02	92.12±0.26	52.48±0.19	89.78±0.22	86.41±0.31	96.08±0.16	88.32±0.26	78.48±0.27	93.78±0.98	55.9±5.55	56.06±0.67	96.35±0.18	70.21±0.81	78.44±0.74	39.15±0.47	36.29±0.39
LP	0.05	89.48±0.11	60.53±0.28	75.71±0.07	99.58±0.01	92.02±0.15	57.44±0.17	55.96±0.15	83.13±0.36	93.59±0.18	82.78±0.3	75.74±0.0	55.39±0.1	37.69±0.04	43.1±0.07	26.01±0.06	37.72±0.03	67.23±0.71	19.94±0.1	27.71±0.17
Bias	0.18	89.14±0.75	61.38±0.31	76.33±0.04	99.65±0.02	90.64±0.97	51.26±0.31	86.38±0.19	85.76±0.32	95.33±0.18	83.71±0.18	77.17±0.35	74.3±1.65	48.27±0.47	52.19±0.48	93.78±1.71	65.5±0.83	75.34±1.26	31.51±0.34	29.18±0.23
VPT	0.10	89.79±0.46	57.8±0.23	73.46±0.22	99.58±0.03	92.3±0.22	55.55±0.1	58.33±0.24	83.11±0.16	93.13±0.2	83.01±0.12	74.76±0.38	58.58±0.45	46.52±0.76	39.0±0.47	53.09±0.52	27.38±3.56	64.93±0.43	20.75±0.46	31.44±1.06
Conv. Par.	7.83	90.94±0.32	66.0±0.06	74.91±0.44	98.81±0.21	92.4±0.18	52.87±0.26	88.44±0.46	85.96±0.17	95.61±0.08	85.72±0.25	77.86±0.11	86.53±1.66	59.48±1.19	55.0±0.19	93.67±0.65	67.11±0.78	83.5±0.88	39.01±0.21	34.72±0.21
Conv. Seq.	9.58	90.28±0.31	68.28±0.94	76.22±0.54	98.48±0.09	91.29±0.08	53.43±0.27	88.03±0.25	86.32±0.05	94.98±0.24	85.64±0.18	77.69±0.16	91.17±0.7	51.15±6.0	52.88±0.44	90.58±0.79	68.22±0.24	83.08±0.83	38.26±0.66	37.41±0.79
Res. Par.	9.14	91.41±0.9	64.98±0.25	73.33±0.5	99.43±0.02	91.66±0.28	52.21±0.21	88.94±0.38	85.59±0.34	95.51±0.13	84.51±0.22	77.58±0.21	89.23±0.41	56.34±0.99	55.14±0.13	90.88±0.1	65.65±0.52	81.25±1.19	38.1±0.28	37.78±0.4
Res. Seq.	10.73	89.26±1.24	63.75±0.76	74.61±0.33	99.33±0.11	90.69±0.34	51.37±0.38	88.47±0.45	85.77±0.27	95.57±0.13	85.47±0.7	77.72±0.12	91.54±0.47	52.26±1.89	55.06±0.51	61.9±1.12	64.35±0.49	82.93±0.07	36.74±0.2	38.72±1.47

8.2. Detailed Results on FGVC

We provide the per-task results on FGVC on ResNet50 BiT-M [27] and ConvNext-B [36] in Tab. 13 and Tab. 14 respectively.

Table 13. Per-task FGVC results of ImageNet-21k pre-trained ResNet50 BiT-M.

Tuning	# Param	CUB200	Stanford Dogs	Stanford Cars	NABirds
FT	24.15	84.51±0.08	79.75±0.08	89.59±0.25	79.97±0.15
LP	0.63	86.07±0.13	80.48±0.07	64.31±0.26	70.89±0.02
Bias	0.67	79.13±0.28	76.49±0.11	34.63±0.1	69.68±0.11
VPT	0.69	85.96±0.1	79.58±0.11	56.9±0.46	76.72±0.1
Conv. Par.	1.22	86.41±0.2	82.07±0.1	85.78±0.25	80.83±0.09
Conv. Seq.	1.06	85.48±0.19	80.5±0.09	73.47±11.62	79.27±0.15
Res. Par.	7.82	85.98±0.15	81.91±0.11	88.96±0.05	80.13±0.16
Res. Seq.	11.80	85.85±0.22	80.69±0.01	87.59±0.16	79.68±0.12

Table 14. Per-task FGVC results of ImageNet-21k pre-trained ConvNext-B.

Tuning	# Param	CUB200	Stanford Dogs	Stanford Cars	NABirds
FT	87.87	89.31±0.18	87.18±0.07	93.43±0.24	88.01±0.17
LP	0.31	90.46±0.02	89.86±0.1	74.96±0.06	85.76±0.02
Bias	0.44	90.86±0.07	89.46±0.03	92.05±0.12	88.25±0.04
VPT	0.37	89.83±0.02	89.95±0.12	74.64±0.06	85.69±0.05
Conv. Par.	5.81	89.83±0.22	88.38±0.34	91.83±0.18	87.06±0.07
Conv. Seq.	3.11	76.5±18.24	86.77±0.28	91.32±0.23	87.4±0.05
Res. Par.	5.73	90.09±0.08	88.06±0.18	90.78±0.14	86.53±0.06
Res. Seq.	8.04	88.57±0.07	87.68±0.07	91.61±0.1	87.03±0.04