
Supplementary Materials of “Correlated Low-Rank Adaptation for ConvNets”

Anonymous Author(s)

Affiliation

Address

email

1	Contents	
2	A Proof of Lemma 3.1	2
3	B More Evidence on Weight Update Correlation in ConvNets	2
4	C Details of Filtering-based Strategy	4
5	D Computational Complexity Analysis	5
6	E Details of Selected Pre-trained Backbones	6
7	F Training Configurations	6
8	G More Details on VTAB-1k Benchmark	8
9	H Limitations	8
10	I Scaling CoLoRA on ADE20K	9

11 A Proof of Lemma 3.1

12 Suppose all elements in $A \in \mathbb{R}^{d \times r}$ are i.i.d., initialized with the uniform distribution $\mathcal{U}(-a, a)$.
 13 Typically, we have $\mathbb{E}[A_{ij}] = 0$ and $\text{Var}[A_{ij}] = \mathbb{E}[A_{ij}^2] = \frac{1}{3}a^2$. Notably, we have $\sigma_{A_{ij}}^2 = \frac{1}{3}a^2$.
 14 Denote the matrix multiplication result as $T = AA^T$; then we can calculate

$$\mathbb{E}[T_{ij}] = \mathbb{E}\left[\sum_{k=1}^r A_{ik}A_{jk}\right] = \sum_{k=1}^r \mathbb{E}[A_{ik}A_{jk}] = \begin{cases} 0 & i \neq j, \\ r\sigma_A^2 & \text{else} \end{cases} \quad (1)$$

15 Therefore, we have $\mathbb{E}[AA^T] = r\sigma_A^2 \mathbf{I}_d$, where \mathbf{I}_d denotes the $d \times d$ identity matrix. Next, we analyze
 16 the variance of the elements in T . For the off-diagonal elements of T , we have

$$\text{Var}[T_{ij}] = \sum_{k=1}^r \mathbb{E}[A_{ik}^2 A_{jk}^2] = r\sigma_A^4, s.t., i \neq j. \quad (2)$$

17 For the diagonal elements, we have

$$\text{Var}[T_{ii}] = \sum_{k=1}^r \mathbb{E}[A_{ik}^4] - \mathbb{E}[A_{ik}^2]^2 = \frac{4}{5}r\sigma_A^4, \quad (3)$$

18 where we can compute $\mathbb{E}[A_{ik}^4] = \frac{1}{a} \int_0^a x^4 dx = \frac{1}{5}a^4 = \frac{9}{5}\sigma_A^4$. Summarizing the above results, we
 19 obtain the variance of AA^T as

$$\text{Var}[T_{ij}] = \begin{cases} r\sigma_A^4 & i \neq j, \\ \frac{4}{5}r\sigma_A^4 & \text{else.} \end{cases} \quad (4)$$

20 In this paper, we approximate $\text{Var}[T] \approx r\sigma_A^4 \mathbf{1}_d$, where $\mathbf{1}_d$ means a $d \times d$ matrix with all elements
 21 being 1. The approximation error in terms of the Frobenius norm can be formed as

$$\text{Err} = \|\text{Var}[T] - r\sigma_A^4 \mathbf{1}_d\|_F^2 = \frac{1}{25}r^2 d \sigma_A^8. \quad (5)$$

22 Note that with kaiming_uniform initialization where $a \propto 1/\sqrt{d}$, the approximation error $\text{Err} \propto$
 23 r^2/d^3 . Hence, we have

$$\frac{\text{Err}}{\|\text{Var}[T]\|_F^2} \approx \frac{1}{25} \frac{r^2/d^3}{d^2 r^2/d^4} = \frac{1}{25d}. \quad (6)$$

24 It is evident that $\text{Err}/\|\text{Var}[T]\|_F^2$ is independent of the rank r and is inversely proportional to the
 25 channel dimension d . When d is large (e.g., 96, 192, 384, 768 for ConvNeXt-B), we can approximate
 26 $\text{Var}[AA^T] \approx r\sigma_A^4$ with a maximum relative error of $1/(25 \times 96) \approx 0.04\%$.

27 B More Evidence on Weight Update Correlation in ConvNets

28 In this section, we provide additional evidence to investigate the correlation between adjacent
 29 convolution layers. Specifically, we select the ConvNeXt backbone and examine the correlation
 30 between the adjacent pwconv1 and pwconv2 layers. We denote their convolution weights as W_1 and
 31 W_2 , respectively. To demonstrate that the updates of W_1 and W_2 are highly correlated, we compute
 32 ΔW by subtracting the old weight W_{old} from the new weight W_{new} , i.e., $\Delta W = W_{\text{new}} - W_{\text{old}}$.
 33 Typically, W_{old} and W_{new} correspond to the weights saved from the previous and current epochs,
 34 respectively. For ADE20K, model weights are checkpointed every 16K iterations, while for MS-
 35 COCO, weights are saved after every epoch.

36 Since both W_1 and W_2 exhibit full-rank update behavior, we assess their correlation by computing
 37 the mean of the singular values of ΔW_1 and ΔW_2 . If their mean singular values exhibit similar
 38 trends throughout the fine-tuning process, it indicates that the update weight matrices for W_1 and W_2
 39 are highly correlated.

40 Figure 1 presents the mean singular values of ΔW_1 and ΔW_2 throughout the fine-tuning process
 41 on ADE20K. Specifically, we analyze the weights of the pwconv1 and pwconv2 layers across all
 42 residual blocks in ConvNeXt-B. It can be observed that nearly all adjacent pwconv1 and pwconv2
 43 layers exhibit strong correlation in their update behavior.

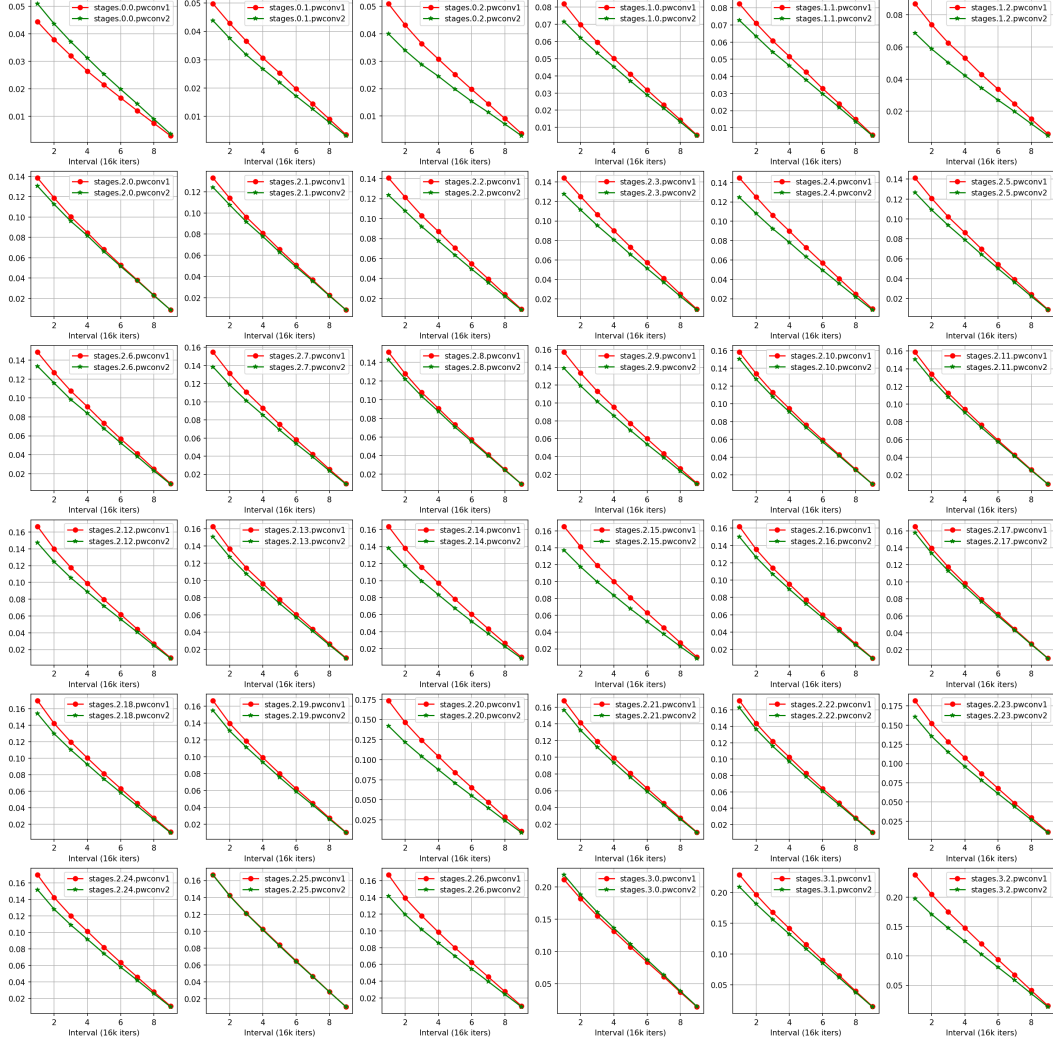


Figure 1: Correlation between pwconv1 and pwconv2 layers in all 36 residual blocks. We fully fine-tune the pre-trained ConvNeXt-B on ADE20K.

Figure 2 shows the mean singular values of ΔW_1 and ΔW_2 throughout the fine-tuning process on MS-COCO. Specifically, we analyze the weights of the pwconv1 and pwconv2 layers across all residual blocks in ConvNeXt-S. It can be observed that nearly all adjacent pwconv1 and pwconv2 layers exhibit strong correlation. The sharp drop in mean singular values is attributed to the learning rate scaling in the standard 1x schedule. The results in Figures 1 and 2 collectively demonstrate that adjacent convolution layers exhibit high correlation during fine-tuning on downstream tasks such as semantic segmentation and object detection.

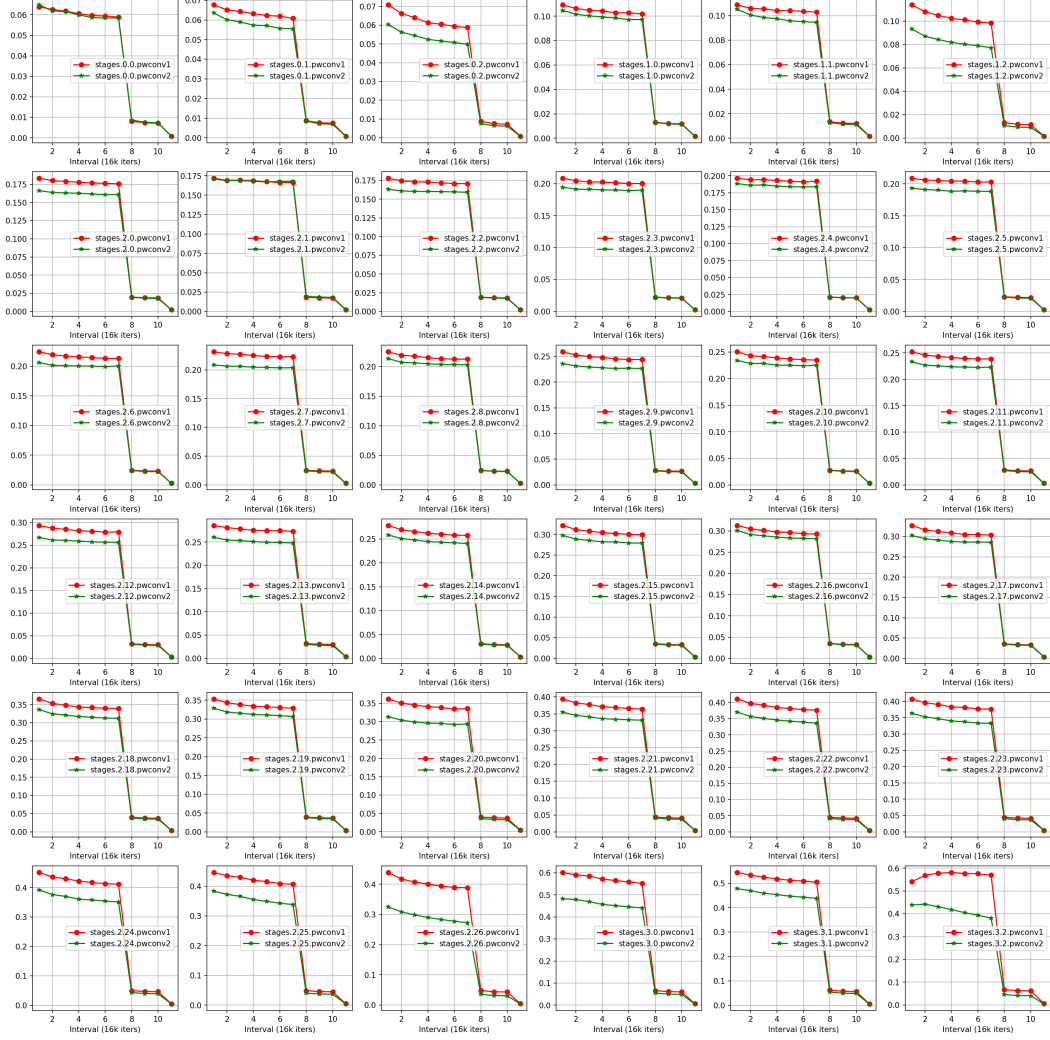


Figure 2: Correlation between pwconv1 and pwconv2 layers in all 36 residual blocks. We fully fine-tune the pre-trained ConvNeXt-S on MS-COCO.

51 C Details of Filtering-based Strategy

52 We first recall in Section 3.4 that the filtering process can be formulated as

$$\tilde{F}_{c,h,w} = \frac{1}{Z_{c,h,w}} \sum_{(\delta h, \delta w) \in \Omega} k_s(\delta h, \delta w) k_r(F_{c,h,w}, F_{c,h+\delta h,w+\delta w}) F_{c,h+\delta h,w+\delta w}, \quad (7)$$

53 where $k_s(\delta h, \delta w)$ denotes a spatial kernel, whereas $k_r(F_{c,h,w}, F_{c,h+\delta h,w+\delta w})$ represents a range
 54 kernel in the context of image filtering. Without the range kernel k_r , the above Eq. (7) can be
 55 simplified as

$$\tilde{F}_{c,h,w} = \frac{1}{Z_{c,h,w}} \sum_{(\delta h, \delta w) \in \Omega} k_s(\delta h, \delta w) F_{c,h+\delta h,w+\delta w} = \left[\text{Conv}\left(\frac{k_s}{Z_c}, F_c\right) \right]_{h,w}, \quad (8)$$

56 where the normalization factor $Z_c = \sum_{\delta h, \delta w} k_s(\delta h, \delta w)$ is spatially invariant. $\text{Conv}(K, X)$ means a
 57 convolution operation on X utilizing a kernel K . Eq. (8) indicates that a channel-wise filtering on
 58 deep features can be expressed as a depthwise convolution operation, which can be rapidly computed
 59 with the help of existing algorithm.

60 However, with the range kernel $k_r(\cdot, \cdot)$, Eq. (7) requires a double-loop process through all (h, w)
 61 coordinates. In this paper, we utilize a special range kernel $k_r(x, y) = \cos(\gamma_c(x - y)) =$

62 $\cos(\gamma_c x) \cos(\gamma_c y) + \sin(\gamma_c x) \sin(\gamma_c y)$, where γ_c is a constant to constrain that $\gamma_c(x - y) \in$
63 $[-\pi/2, \pi/2]$. Incorporating this range kernel into Eq. (7), we derive

$$\begin{aligned}\tilde{F}_{c,h,w} &= \frac{1}{Z_{c,h,w}} \sum_{(\delta h, \delta w) \in \Omega} k_s(\delta h, \delta w) k_r(F_{c,h,w}, F_{c,h+\delta h, w+\delta w}) F_{c,h+\delta h, w+\delta w}, \\ &= \frac{1}{Z_{c,h,w}} \sum_{(\delta h, \delta w) \in \Omega} k_s(\delta h, \delta w) \cos \gamma_c F_{c,h,w} \cos \gamma_c F_{c,h+\delta h, w+\delta w} F_{c,h+\delta h, w+\delta w} \\ &\quad + \frac{1}{Z_{c,h,w}} \sum_{(\delta h, \delta w) \in \Omega} k_s(\delta h, \delta w) \sin \gamma_c F_{c,h,w} \sin \gamma_c F_{c,h+\delta h, w+\delta w} F_{c,h+\delta h, w+\delta w} \\ &= \frac{[\cos \gamma_c F \odot \text{Conv}(k_s, \cos \gamma_c F \odot F)]_{h,w} + [\sin \gamma_c F \odot \text{Conv}(k_s, \sin \gamma_c F \odot F)]_{h,w}}{Z_{c,h,w}}, \quad (9)\end{aligned}$$

64 where the \odot means the element-wise multiplication. Specifically, the normalization factor is

$$\begin{aligned}Z_{c,h,w} &= \sum_{(\delta h, \delta w) \in \Omega} k_s(\delta h, \delta w) \cos \gamma_c F_{c,h,w} \cos \gamma_c F_{c,h+\delta h, w+\delta w} \\ &\quad + \sum_{(\delta h, \delta w) \in \Omega} k_s(\delta h, \delta w) \sin \gamma_c F_{c,h,w} \sin \gamma_c F_{c,h+\delta h, w+\delta w} \\ &= [\cos \gamma_c F \odot \text{Conv}(k_s, \cos \gamma_c F) + \sin \gamma_c F \odot \text{Conv}(k_s, \sin \gamma_c F)]_{h,w}. \quad (10)\end{aligned}$$

65 Combining Eqs. (9) and (10), we obtain the final simplified form of Eq. (7) as follows:

$$\tilde{F}_c = \frac{\cos \gamma_c F \odot \text{Conv}(k_s, \cos \gamma_c F \odot F) + \sin \gamma_c F \odot \text{Conv}(k_s, \sin \gamma_c F \odot F)}{\cos \gamma_c F \odot \text{Conv}(k_s, \cos \gamma_c F) + \sin \gamma_c F \odot \text{Conv}(k_s, \sin \gamma_c F) + \epsilon}, \quad (11)$$

66 where we set $\epsilon = 10^{-5}$ to avoid division by zero. As shown in Eq. (11), only four depthwise
67 convolution operations are required to compute Eq. (7), eliminating the need for time-consuming
68 for-loops. For implementation, we employ a fixed Gaussian spatial kernel k_s with a window size of
69 11 and a standard deviation of $\sigma = 1.0$. Consequently, Eq. (11) contains no trainable parameters, and
70 gradients need not be computed for this filtering process.

Table 1: Complexity analysis of different LoRA methods. Here, r denotes the rank of matrices A and B , #Params represents the number of trainable parameters in LoRA matrices, and FLOPs measure the computational complexity of LoRA modulation during both training and inference phases.

Method	Rank r	#Params	FLOPs (train)	FLOPs (inference)
LoRA	r	$r(d + 2d' + d'')$	$(hwd' + rd' + d')(d + d'')$	$hwd'(d + d'')$
PiSSA	r	$r(d + 2d' + d'')$	$(hwd' + rd' + d')(d + d'')$	$hwd'(d + d'')$
HiRA	r	$r(d + 2d' + d'')$	$(hwd' + rd' + 2d')(d + d'')$	$hwd'(d + d'')$
CoLoRA (ours)	$r_l + r_s = r$	$r(d + d'') + 2r_l d'$	$(hwd' + rd' + 2d' + r_s d')(d + d'')$	$hwd'(d + d'')$

71 D Computational Complexity Analysis

72 The computational complexity of CoLoRA stems from both the proposed correlated low-rank
73 adaptation process and the filtering technique.

74 **Complexity of correlated low-rank adaptation.** We analyze the complexity of two sequential 1×1
75 convolutional layers as described in Section 3. This analysis can be extended to arbitrary convolution
76 layers. Consider an input feature $X \in \mathbb{R}^{h \times w \times d}$ (with spatial dimensions rearranged for simplicity),
77 where d denotes the channel dimension and h, w represent its spatial resolution. The feature X is
78 processed by two weight matrices $W_1 \in \mathbb{R}^{d \times d'}$ and $W_2 \in \mathbb{R}^{d' \times d''}$. For clarity, we omit the analysis
79 of bias terms as they are independent of LoRA. The vanilla LoRA modifies the feature X through

$$Y_1 = X(W_1 + s_1 A_1 B_1), \quad (12)$$

$$Y_2 = Y_1(W_2 + s_2 A_2 B_2), \quad (13)$$

where $A_1 \in \mathbb{R}^{d \times r}$, $B_1 \in \mathbb{R}^{r \times d'}$, $A_2 \in \mathbb{R}^{d' \times r}$, and $B_2 \in \mathbb{R}^{r \times d''}$ are low-rank matrices with rank r , while s_1 and s_2 denote scaling factors. We contend that such layer-independent LoRA formulations fail to capture inter-layer correlations, which are particularly crucial for effectively fine-tuning convolutional networks. To overcome this limitation, we propose CoLoRA, which transforms the feature X through

$$Y_1 = X(W_1 + s_1 A_1 B_1 + s A_s B_s W_2^T) \quad (14)$$

$$Y_2 = Y_1(W_2 + s_2 A_2 B_2 + s W_1^T A_s B_s), \quad (15)$$

where we incorporate shared low-rank matrices $A_s \in \mathbb{R}^{d \times r_s}$ and $B_s \in \mathbb{R}^{r_s \times d''}$ to explicitly model correlations between adjacent layers. To efficiently compute $A_s B_s W_2^T$ in Eq. (15), we first compute $C = B_s W_2^T$ followed by the product $A_s C$, leveraging the low-rank property $r_s \ll \min\{d, d', d''\}$. The computation of $W_1^T A_s B_s$ follows an analogous procedure. Table 1 compares the computational complexity of LoRA, PiSSA, HiRA, and our proposed CoLoRA.

During training, CoLoRA introduces an additional computational cost of $r_s d'(d + d'')$ FLOPs compared to existing HiRA methods. Note that since $r_s \ll \min\{hw, d, d', d''\}$, this overhead is negligible in practice. Furthermore, due to CoLoRA’s weight decomposition formulation, we can directly incorporate the weight update matrices into the base weights W_1 and W_2 after training completion. Consequently, CoLoRA introduces no additional computational overhead during inference.

Complexity of filtering technique. As demonstrated in Eq. (11), the filtering operation can be efficiently implemented using four depthwise convolutions. To optimize computational efficiency, we apply the filtering process exclusively to features compressed by the LoRA matrix A . This design yields an overall computational complexity of $O(hwk^2r)$, where k denotes the window size ($k = 11$ in our implementation).

E Details of Selected Pre-trained Backbones

In Table 2, we present the specifications of the pre-trained ConvNets backbones used in our study.

Table 2: Details of selected pre-trained backbones in this paper.

Backbones	Pre-train dataset	#Params	Feature dim.	Public url
ResNet-50	ImageNet-1K	23.5M	2048	https://docs.pytorch.org/vision/main/models.html (v2)
ConvNeXt-S	ImageNet-22K	49.5M	768	https://dl.fbaipublicfiles.com/convnext/convnext_small_22k_1k_384.pth
ConvNeXt-B	ImageNet-22K	87.6M	1024	https://dl.fbaipublicfiles.com/convnext/convnext_base_22k_224.pth

F Training Configurations

We conduct experiments by training ResNet-50, ConvNeXt-S, and ConvNeXt-B models on multiple downstream vision tasks. The detailed training configurations are provided in this section. The basic experimental configurations employed in our study are summarized in Tables 3 and 4. Notably, our implementation follows the official configurations from <https://github.com/facebookresearch/ConvNeXt>.

Basic configuration. The baseline configuration employs consistent hyperparameters across different downstream tasks and PEFT methods (with potential exceptions for MS-COCO, as detailed below). Tables 3 and 4 present the complete specifications of baseline hyperparameters and training configurations.

Training configuration on VTAB-1k. For VTAB-1k classification tasks, our processing pipeline consists of a Global Average Pooling (GAP) layer, followed by normalization and a linear classifier head. We optimize the models using standard cross-entropy loss. Each VTAB-1k sub-task is fine-tuned with a batch size of 32 for 100 epochs. In addition to the trainable low-rank matrices, we optimize all bias terms and normalization layers in the backbone network. Unlike existing approaches such as Conv-Adapter that perform grid search to identify optimal hyperparameters for individual sub-tasks, we maintain consistent hyperparameter settings across all sub-tasks. The specific configurations for each PEFT method are detailed below:

Table 3: Basic configuration for ResNet-50.

Hyper-parameter	Setting
Optimizer	AdamW
Learning rate	1e-4
Weight decay	0.05
Layer weight decay	Number of layers is 16. Decay rate is 0.99
Linear warmup	1500 steps with ratio 1e-6
Mixed precision training	APEX

Table 4: Basic configuration for ConvNeXt (ConvNeXt-S and ConvNeXt-B).

Hyper-parameter	Setting
Optimizer	AdamW
Learning rate	1e-4
Weight decay	0.05
Stage-wise decay	Number of stages is 4. Decay rate is 0.9
Linear warmup	1500 steps with ratio 1e-6
Mixed precision training	APEX

- **Conv-Adapter.** We utilize the official Conv-Adapter implementation from <https://github.com/Hhhhhhao/Conv-Adapter>. As Conv-Adapter exclusively applies trainable adapters to spatial convolution layers with kernel sizes greater than 1, we set its channel compression factor to $\gamma = 4$ to maintain parameter count comparability with other methods. This configuration results in Conv-Adapter having the highest rank among all compared methods' low-rank matrices.
- **PiSSA.** We employ the official PiSSA implementation from <https://github.com/GraphPKU/PiSSA>. PiSSA is applied to all 1×1 convolutional layers, as directly re-shaping convolution kernels from shape $[C_{out}, C_{in}, k, k]$ to $[C_{out}, C_{in} \times k^2]$ for standard LoRA application would compromise the network's inherent inductive bias. In practice, we set $\gamma = 20$ to maintain parameter count consistency with other methods. Following established LoRA practices, we use a default scaling factor configuration of $\alpha/r = 2$ as recommended in recent literature.
- **HiRA.** We adapt the official HiRA implementation from <https://github.com/hqsiswiliam/hira> for convolutional networks. Following the same approach as with PiSSA, we restrict HiRA application exclusively to 1×1 convolutional layers. To maintain parameter count consistency, we set $\gamma = 20$ and utilize the theoretically derived scaling factors s_{expect} presented in Section 3.1.

Training configuration on ADE20K. For ADE20K segmentation, we employ both a primary decoder head and an auxiliary head. The primary decoder head is optimized using cross-entropy loss (weight = 1.0) and incorporates deep features from all four network stages in both ResNet-50 and ConvNeXt architectures. In contrast, the auxiliary head employs only third-stage features and is trained with a reduced cross-entropy loss weight of 0.4.

For both full fine-tuning and PEFT approaches, we employ the standard 160k iteration schedule. Using a batch size of 16, we perform evaluations at 16k-iteration intervals. In addition to the introduced trainable low-rank matrices, we optimize all bias terms and normalization layers. The PEFT method configurations for ADE20K segmentation remain identical to those used for VTAB-1k classification.

Training configuration on MS-COCO. We conduct object detection experiments using the Faster R-CNN framework, adopting the configuration from https://github.com/SwinTransformer/Swin-Transformer-Object-Detection/tree/master/configs/_base_/models. Our implementation differs from Table 4 in three key aspects: (1) a learning rate of 2×10^{-4} , (2) a standard 500-iteration warmup period, and (3) a batch size of 16 distributed across two NVIDIA A800 GPUs. Following the standard 1x training schedule

Table 5: Average top-1 accuracy on the VTAB-1k **NATURAL** benchmark across three independent runs. The highest-performing PEFT method for each task is highlighted in **bold**.

Backbone	Method	#Param	Caltech101	CIFAR-100	DTD	Flowers102	Pets	Sun397	SVHN	Average
ResNet-50	FT	23.5M	89.49±0.19	30.82±0.16	64.41±0.67	89.35±0.34	84.43±0.46	31.00±0.17	82.51±0.03	67.43±0.29
	Conv-Adapter [1]	1.4M	86.98±0.13	27.22±0.32	64.40 ±0.59	82.21±0.19	88.98±0.08	32.67±0.06	51.31±0.78	61.97±0.31
	PiSSA [2]	1.2M	88.12±0.26	26.74±0.49	62.93±0.15	85.85±0.91	89.13±0.05	32.71 ±0.27	63.32±0.30	64.11±0.35
	HiRA [3]	1.2M	87.00±0.18	27.97±0.07	63.99±0.30	82.39±0.26	89.20 ±0.14	32.28±0.14	53.31±0.64	62.31±0.25
	CoLoRA (ours)	1.0M	89.12 ±0.24	29.98 ±0.48	62.82±0.74	87.51 ±0.25	88.87±0.45	32.38±0.57	75.31 ±0.78	66.57 ±0.50
ConvNeXt-S	FT	49.5M	91.33±0.55	65.60±0.32	74.17±0.05	98.74±0.10	90.36±0.22	49.02±0.19	92.30±0.13	80.22±0.22
	Conv-Adapter [1]	2.8M	90.94±0.45	68.52±0.36	75.37±0.40	99.12±0.07	91.13±0.09	49.91±0.06	90.35±0.36	80.76±0.26
	PiSSA [2]	3.0M	90.93±0.12	69.22±0.40	75.62±0.10	99.06±0.04	91.29±0.16	51.96±0.29	90.27±0.29	81.19±0.20
	HiRA [3]	3.0M	90.71±0.24	70.85 ±0.20	76.30 ±0.11	99.27 ±0.00	92.12 ±0.07	53.56 ±0.14	86.39±0.10	81.31 ±0.12
	CoLoRA (ours)	2.6M	91.60 ±0.11	66.34±0.55	75.00±0.60	98.89±0.06	90.43±0.27	49.94±0.33	92.08 ±0.21	80.61±0.31

Table 6: Average top-1 classification accuracy on the VTAB-1k **SPECIALIZED** benchmark across three experimental trials. Top-performing PEFT methods are indicated in **bold**.

Backbone	Method	#Param	Camelyon	EuroSAT	Resisc45	Retinopathy	Average
ResNet-50	FT	23.5M	85.31±0.61	91.83±0.34	80.94±0.11	73.82±0.20	82.98±0.35
	Conv-Adapter [1]	1.4M	78.59±0.28	88.20±0.44	75.29±0.23	73.80±0.06	78.97±0.25
	PiSSA [2]	1.2M	81.59±0.58	90.25±0.55	78.47±0.56	73.82±0.15	81.03±0.46
	HiRA [3]	1.2M	79.07±0.10	89.10±0.03	75.42±0.11	73.85±0.02	79.36±0.07
	CoLoRA (ours)	1.0M	82.60 ±0.58	92.16 ±0.20	81.08 ±0.26	74.30 ±0.19	82.54 ±0.31
ConvNeXt-S	FT	49.5M	87.95±0.45	95.98±0.06	85.98±0.75	76.83±0.27	86.69±0.38
	Conv-Adapter [1]	2.8M	85.75±0.75	94.89±0.16	84.04±0.24	75.30±0.22	84.99±0.34
	PiSSA [2]	3.0M	85.79±0.07	95.45±0.09	85.56±0.15	75.56±0.26	85.59±0.14
	HiRA [3]	3.0M	84.29±0.20	94.85±0.07	84.91±0.06	75.85±0.12	84.97±0.11
	CoLoRA (ours)	2.6M	87.51 ±0.07	96.00 ±0.12	85.95 ±0.15	77.13 ±0.12	86.65 ±0.11

(https://github.com/SwinTransformer/Swin-Transformer-Object-Detection/blob/master/configs/_base_/schedules/schedule_1x.py), we train models on MS-COCO for 12 epochs with learning rate adjustments at the 8th and 11th epochs. The PEFT method configurations for object detection remain consistent with those used for VTAB-1k classification and ADE20K segmentation.

G More Details on VTAB-1k Benchmark

We present comprehensive quantitative results on the VTAB-1k benchmark, including the average top-1 accuracy and its standard deviation in Tables 5 to 7. All methods were evaluated on VTAB-1k across three independent runs. Notably, the proposed CoLoRA significantly outperforms existing PEFT methods in the **NATURAL**, **SPECIALIZED**, and **STRUCTURED** categories with ResNet-50 as the backbone. When using ConvNeXt-S, CoLoRA achieves the highest top-1 accuracy in the **SPECIALIZED** and **STRUCTURED** categories, compared to state-of-the-art PEFT methods. Furthermore, CoLoRA surpasses full fine-tuning in the **STRUCTURED** category, highlighting its superior adaptation capability. Consequently, CoLoRA outperforms full fine-tuning on VTAB-1k with ConvNeXt-S as the backbone.

H Limitations

The proposed CoLoRA introduces a pair of layer-sharing low-rank matrices to capture inter-layer correlations between adjacent convolutional layers. The correlation strength, determined by the rank of the shared matrices, directly controls the number of tunable parameters. Specifically, a higher rank of the sharing matrices results in fewer tunable parameters. Conversely, the correlation strength exhibits an ambiguous influence on adaptation performance, necessitating a hyper-parameter tuning process. In this work, we employ a single hyper-parameter γ to regulate the correlation strength across all layers. Nevertheless, a layer-specific configuration of γ could potentially enhance adaptation performance. It should be noted that the default configuration presented in our paper already demonstrates superior performance compared to state-of-the-art PEFT methods. An additional

Table 7: Average top-1 accuracy across three runs on the VTAB-1k **STRUCTURED** benchmark. The highest-performing PEFT method is highlighted in **bold**.

Backbone	Method	#Param	Clevr-Count	Clevr-Dist	DMLab	dSpr-Loc	dSpr-Ori	KITTI-Dist	sNOB-Azim	sNOB-Elev	Average
ResNet-50	FT	23.5M	43.27±0.58	55.21±0.48	45.67±0.14	80.18±0.82	41.93±0.56	80.59±1.11	26.54±0.16	48.21±1.55	52.70±0.67
	Conv-Adapter [1]	1.4M	35.94±0.05	44.98±1.55	35.40±0.36	41.50±0.62	15.29±0.95	69.95±1.04	14.72±0.12	38.21±0.79	37.00±0.67
	PiSSA [2]	1.2M	48.61±0.29	49.16±0.63	38.21±0.46	52.62±1.11	22.29±0.32	73.70±1.19	18.05±0.61	39.43±0.89	42.76±0.69
	HiRA [3]	1.2M	40.98±0.31	46.69±0.72	35.10±0.40	45.70±0.26	16.94±0.23	71.31±0.75	13.53±0.27	44.05±0.31	39.29±0.41
	CoLoRA (ours)	1.0M	54.04±3.25	53.95±0.86	42.05±0.39	76.42±1.33	37.18±0.53	79.98±0.77	22.58±0.47	48.81±1.23	51.88±1.10
ConvNeXt-S	FT	49.5M	91.06±0.40	66.58±0.60	55.10±0.37	92.75±0.49	62.41±0.77	83.88±0.17	39.96±0.68	46.91±0.18	67.33±0.46
	Conv-Adapter [1]	2.8M	87.82±0.65	66.06±0.62	48.68±0.67	94.11±0.25	61.30±0.47	85.04±0.46	37.29±0.54	49.14±0.45	66.18±0.52
	PiSSA [2]	3.0M	91.51±0.54	67.04±0.90	51.35±0.49	94.47±0.30	61.47±0.65	82.79±0.07	37.24±0.46	49.15±0.05	66.88±0.43
	HiRA [3]	3.0M	79.64±3.01	63.95±0.39	47.33±0.48	79.60±2.20	56.35±0.97	82.32±0.77	25.85±0.25	42.13±0.26	59.65±1.04
	CoLoRA (ours)	2.6M	91.71±0.18	65.57±0.08	54.59±0.61	92.94±0.52	62.28±0.88	83.82±0.40	40.71±0.67	48.95±0.40	67.57±0.47

Table 8: Quantitative evaluation of CoLoRA scaling effects on ConvNeXt-B for ADE20K adaptation.

γ	16	8	8	4	4	4	2	2
$r_s/(r_s + r_l)$	0.5	0.8	0.5	0.8	0.5	0.4	0.9	0.8
#Params	4.6M	5.7M	8.8M	11.1M	17.3M	19.5M	17.7M	21.9M
mIoU	49.55	50.65	50.86	50.86	50.96	51.13	50.38	51.06

179 limitation of this study is that we exclusively evaluated CoLoRA on ConvNets, leaving its efficacy on
180 Transformer-based architectures such as ViT or Swin unexplored.

181 I Scaling CoLoRA on ADE20K

182 In this section, we investigate the relationship between the number of tunable parameters and
183 adaptation performance. Using the ConvNeXt-B backbone, we evaluate different values for γ and the
184 rank ratio $r_s/(r_s + r_l)$. Note that full fine-tuning of ConvNeXt-B on ADE20K achieves 50.99 mIoU.
185 The results, presented in Table 8, demonstrate that CoLoRA with merely 19.5M tunable parameters
186 (compared to 87.6M for full fine-tuning) not only matches but surpasses full fine-tuning performance,
187 reaching 51.13 mIoU. This remarkable performance highlights CoLoRA’s potential for efficiently
188 fine-tuning pre-trained ConvNets across diverse downstream tasks.

189 References

- 190 [1] Hao Chen, Ran Tao, Han Zhang, Yidong Wang, Xiang Li, Wei Ye, Jindong Wang, Guosheng Hu,
191 and Marios Savvides. Conv-adapter: Exploring parameter efficient transfer learning for convnets.
192 In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages
193 1551–1561, 2024.
- 194 [2] Fanxu Meng, Zhaohui Wang, and Muhan Zhang. Pissa: Principal singular values and singular
195 vectors adaptation of large language models. *Advances in Neural Information Processing Systems*,
196 37:121038–121072, 2024.
- 197 [3] Qiushi Huang, Tom Ko, Zhan Zhuang, Lilian Tang, and Yu Zhang. Hira: Parameter-efficient
198 hadamard high-rank adaptation for large language models. In *Proceedings of the International
199 Conference on Learning Representations*, 2025.