Appendix

Overview

This appendix contains additional details of the paper '*Split Adaptation for Pre-trained Vision Transformers*', including more implementation details of the main experiments, additional experiment results, and sensitivity analysis of parameters used in our methods.

- Section A provides more implementation details and setups of our SA (Section A.1) and comparison state-of-the-art baseline approaches (Section A.2).
- Section B reports additional experiment results of performing adaptation on more ViT architectures: Deit-Base and Swin-L.
- Section C shows the sensitivity analysis on the bi-level noise degree (Section C.1) and patch retrieval augmentation (Section C.2) in our SA.

A. More Implementation Details

A.1. More setups of SA

In addition to the setups mentioned in the main paper, we provide more setups of SA here. In the OOD quantization-aware tuning, we cut the merged server dataset into $M = 3$ subsets and quantize $M = 3$ frontends accordingly. As for the random seeds, we randomly use 1, 42, 215 to run experiments repeatedly. Moreover, in the risk assessment of the model protection provided by SA, we adopt the last 1/3 layers of DinoV2-Large [4] as the auxiliary backend.

A.2. Implementation of Baseline Approaches

All baseline approaches along with our SA adopt the same task module, i.e., two linear layers for the Places365 and a single linear layer for other datasets. The batch size we used is 32 and we set the training epochs as when the training loss does not decrease. We follow the default setups of the baseline approaches when setting the optimizer and its learning rate.

Linear Probing [29]: Following the standard setups, we freeze the pre-trained ViT and only tune the task module in 100 epochs.

Fine Tuning: After attaching the task module behind the pre-trained ViT, we forward the client data to the model and tune it entirely in 100 epochs.

LN-TUNE [2]: LN-TUNE only tunes the LayerNorm parameters in transformers, thus we filter out these parameters and tune them along with the task module in 100 epochs.

Split Learning [44]: The same splitting location as our SA is adopted here, i.e., cut the first 2/3 layers of the pre-trained ViT as the frontend, while the rest 1/3 layers form the backend. The client manages the frontend and the task module, while the server hosts the backend. Following the default settings, the backend is frozen, and only the frontend and the task module are optimized by back-propagating the training loss in 100 epochs.

Offsite Tuning [49]: Following the default setups, we adopt the layer-drop strategy to implement Offsite Tuning. Specifically, we regard the first and last two layers of the pre-trained ViT as the adapters that the client will tune. Then the emulator is formed by randomly dropping half intermediate layers (except the layers selected as the adapters). The adapter along with the task module is tuned in 50 epochs on the client side.

Data Reconstruction Attack—FORA [53]: Following the training workflow in FORA [53], we use the frontend model from the server side as the substitute model. In FORA, Multi-Kernel Maximum Mean Discrepancy (MK-MMD) and discriminator losses are used to align the attacker encoder with the client encoder. Since the client does not share its encoder with the server, we designed to use the server frontend directly as the attacker encoder. During training, the frontend is frozen, thus these losses are no longer necessary. We remove the MK-MMD and discriminator during training. We developed an inverse model using decoder layers similar to the pre-trained ViT encoder layers, which are trained to reconstruct data from the bi-level noisy representations.

B. Additional Experiments

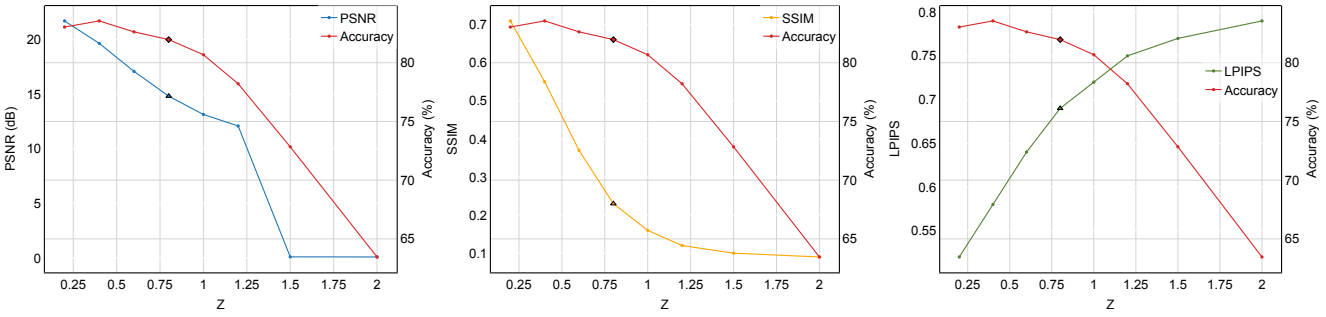
In this section, we experimented with more model architectures: Deit-Base [43] and Swin-L [33]. For Deit-Base, we also split the first 2/3 layers as the frontend and view the rest 1/3 layers as the backend. As for Swin-L, it consists of four stages, and each stage forms a sub-transformer with unique setups of embedding dimensions and attention heads. Therefore, we cut the Swin-L stage-wise, i.e., splitting the first three stages as the frontend while the last stage is the backend. Besides, due to

Table 7. Performance comparison between our SA and other baseline approaches in 3-shot, 5-shot, and 10-shot adaptation scenarios on Deit-Base. The SA can substantially exceed other methods. We bold and blue **the best**, and bold **the second best**.

Few-shot Setup	3-shot			5-shot			10-shot		
Method	CIFAR-100	Places365	D.Net-Cl	CIFAR-100	Places365	D.Net-Cl	CIFAR-100	Places365	D.Net-Cl
Linear Probing	**50.58**$_{\pm 1.93}$	**19.13**$_{\pm 0.53}$	**34.65**$_{\pm 1.67}$	**56.80**$_{\pm 0.87}$	**22.22**$_{\pm 0.12}$	**42.19**$_{\pm 0.28}$	**63.24**$_{\pm 0.80}$	26.99$_{\pm 0.11}$	50.31$_{\pm 1.07}$
Fine Tuning	24.45$_{\pm 2.91}$	15.28$_{\pm 0.10}$	16.25$_{\pm 0.83}$	36.24$_{\pm 2.31}$	20.60$_{\pm 0.75}$	32.36$_{\pm 1.42}$	57.54$_{\pm 1.62}$	**28.05**$_{\pm 0.70}$	52.38$_{\pm 1.66}$
LN-TUNE	11.17$_{\pm 1.71}$	0.96$_{\pm 0.28}$	2.98$_{\pm 0.23}$	16.20$_{\pm 2.02}$	1.55$_{\pm 0.45}$	4.72$_{\pm 0.66}$	18.82$_{\pm 5.58}$	1.86$_{\pm 1.40}$	7.34$_{\pm 0.96}$
Split Learning	20.55$_{\pm 3.68}$	16.32$_{\pm 0.82}$	20.55$_{\pm 0.78}$	30.37$_{\pm 2.07}$	20.79$_{\pm 0.96}$	**42.19**$_{\pm 0.28}$	42.94$_{\pm 0.58}$	26.14$_{\pm 1.05}$	48.33$_{\pm 3.03}$
Offsite Tuning	26.18$_{\pm 1.99}$	15.64$_{\pm 1.67}$	21.40$_{\pm 1.16}$	37.20$_{\pm 3.06}$	21.19$_{\pm 0.41}$	33.42$_{\pm 2.20}$	58.14$_{\pm 2.40}$	27.84$_{\pm 0.60}$	**52.58**$_{\pm 0.74}$
SA (ours)	**48.86**$_{\pm 1.60}$	**21.70**$_{\pm 0.52}$	**34.55**$_{\pm 1.49}$	**58.22**$_{\pm 0.79}$	**26.16**$_{\pm 0.16}$	**46.64**$_{\pm 0.34}$	**68.90**$_{\pm 0.57}$	**31.75**$_{\pm 0.31}$	**58.27**$_{\pm 0.49}$

Table 8. Performance comparison between our SA and other baseline approaches in 3-shot, 5-shot, and 10-shot adaptation scenarios on Swin-L. The SA can substantially exceed other methods. We bold and blue **the best**, and bold **the second best**.

Few-shot Setup	3-shot			5-shot			10-shot		
Method	CIFAR-100	Places365	D.Net-Cl	CIFAR-100	Places365	D.Net-Cl	CIFAR-100	Places365	D.Net-Cl
Linear Probing	**71.28**$_{\pm 1.79}$	**33.17**$_{\pm 0.54}$	**58.22**$_{\pm 0.80}$	**75.76**$_{\pm 1.07}$	**36.95**$_{\pm 0.36}$	64.76$_{\pm 0.79}$	79.83$_{\pm 0.40}$	**40.62**$_{\pm 0.25}$	70.40$_{\pm 0.26}$
Fine Tuning	61.23$_{\pm 2.53}$	29.12$_{\pm 0.42}$	55.74$_{\pm 0.64}$	73.05$_{\pm 1.23}$	35.09$_{\pm 0.32}$	**65.26**$_{\pm 1.63}$	**82.65**$_{\pm 0.41}$	39.79$_{\pm 0.17}$	**72.92**$_{\pm 1.06}$
LN-TUNE	28.98$_{\pm 7.47}$	4.32$_{\pm 1.62}$	7.32$_{\pm 3.20}$	29.23$_{\pm 16.2}$	6.35$_{\pm 1.64}$	12.45$_{\pm 3.63}$	27.42$_{\pm 11.8}$	8.12$_{\pm 2.24}$	17.72$_{\pm 11.0}$
Split Learning	51.51$_{\pm 4.66}$	25.76$_{\pm 0.22}$	49.64$_{\pm 2.66}$	65.37$_{\pm 1.41}$	32.19$_{\pm 0.43}$	59.36$_{\pm 2.73}$	73.59$_{\pm 2.57}$	38.30$_{\pm 0.44}$	70.33$_{\pm 1.38}$
Offsite Tuning	62.23$_{\pm 2.27}$	29.66$_{\pm 0.51}$	55.03$_{\pm 0.87}$	72.80$_{\pm 0.65}$	35.75$_{\pm 0.49}$	65.20$_{\pm 0.90}$	80.68$_{\pm 0.83}$	39.71$_{\pm 0.83}$	71.36$_{\pm 1.05}$
SA (ours)	**70.22**$_{\pm 0.92}$	**31.87**$_{\pm 0.31}$	**59.83**$_{\pm 1.61}$	**76.66**$_{\pm 0.75}$	**37.20**$_{\pm 0.28}$	**66.13**$_{\pm 0.92}$	**80.96**$_{\pm 0.28}$	**41.00**$_{\pm 0.44}$	**72.13**$_{\pm 0.59}$



(a) PSNR (b) SSIM (c) LPIPS

Figure 4. Sensitivity analysis of bi-level noisy representation extraction by changing the injected Laplace noise degree with a variety of z.

Table 9. Sensitivity analysis of patch number N^{P} in our patch retrieval augmentation when setting the augmentation runs as $N^{\mathrm{Aug}} = 64$.

N^{P}	0	10	20	30	40	50	60	70	80	90	100	120	140	160	180
SA	80.41	81.60	80.61	80.88	80.61	82.52	80.11	82.33	80.78	82.18	81.54	81.86	81.84	81.92	81.52

Table 10. Sensitivity analysis of augmentation runs N^{Aug} in our patch retrieval augmentation when setting the patch number as $N^{\mathrm{P}} = 50$.

N^{Aug}	0	1	2	4	8	20	30	40	50	60	64	70	80	90	100
SA	80.41	79.94	81.02	81.83	81.95	81.41	81.24	80.83	80.85	81.15	82.52	81.14	81.49	82.26	80.84

the window sliding mechanism in the swin-transformer, the patch sequence of each sample is uncertain, making our patch retrieval augmentation inapplicable. The setups of the task module are the same as those of ViT-L. The adaptation scenarios are still 3-shot, 5-shot, and 10-shot. According to the experimental results shown in Tables 7 and 8, we can clearly observe that our SA approach achieves the best performance in almost all cases, in particular, note that SA does not apply patch retrieval augmentation in experiments of Swin-L. Methods whose tuning spreads over the entire model, like Fine Tuning, LN-TUNE, and Offsite Tuning, perform much poorer than others. The potential reason is still the overfitting of the limited client data. As for the partial tuning methods, like Linear Probing and Split Learning, they are influenced much less by overfitting but still lag far behind our SA.

C. Sensitivity Analysis

C.1. Analysis of Bi-level Noise Degree

In SA, the bi-level noise (Section 3.4) directly impacts the adaptation performance and the defensive capability regarding data reconstruction attacks. As a result, we conduct a sensitivity analysis on CIFAR-100 by changing the noise degree. Specifically, we adopt a series of Laplace noises—$\mathrm{Laplace}(0, 0.2)$, $\mathrm{Laplace}(0, 0.4)$, $\mathrm{Laplace}(0, 0.6)$, $\mathrm{Laplace}(0, 0.8)$, $\mathrm{Laplace}(0, 1.0)$, $\mathrm{Laplace}(0, 1.2)$, $\mathrm{Laplace}(0, 1.5)$, and $\mathrm{Laplace}(0, 2.0)$—to conduct the pre-trained ViT adaptation and then launch FORA to reconstruct the client data. We calculate the three metrics to measure the reconstruction quality and associate them with the adaptation performance to draw the trade-off curves in Figure 4. According to these experimental results, the noise $\mathrm{Laplace}(0, 0.8)$ achieves a good trade-off between adaptation performance and client data protection.

C.2. Analysis of Patch Retrieval Augmentation

The patch retrieval augmentation (Section 3.5) in SA requires two hyperparameters: one is the number of the patches N^{P} that need retrieval and replacement, the other is the number of augmentation runs N^{Aug}. We conduct a sensitivity analysis of N^{P} and N^{Aug} on CIFAR-100. We first empirically set $N^{\mathrm{Aug}} = 64$ and adopt a series of N^{P}s. Then we select a patch number $N^{\mathrm{P}} = 50$ that performs relatively well and change the N^{Aug} to conduct experiments. The results are shown in Tables 9 and 10, in which we can observe that no matter what setups of N^{P} and N^{Aug}, our patch retrieval augmentation is generally effective in enhancing the adaptation though there is certain fluctuation. Then we adopt an empirically good setup ($N^{\mathrm{P}} = 50$ and $N^{\mathrm{Aug}} = 64$) in our main experiments.