



HIGH-EFFICIENT DIFFUSION MODEL FINE-TUNING WITH PROGRESSIVE SPARSE LOW-RANK ADAPTATION (APPENDIX)

Teng Hu^{1*}, Jiangning Zhang^{2*}, Ran Yi^{1†}, Hongrui Huang¹, Yabiao Wang², Lizhuang Ma¹

¹ Shanghai Jiao Tong University ² YouTu Lab, Tencent
 {hu-teng, ranyi, lzma}@sjtu.edu.cn 2022212016@stu.hit.edu.cn
 {vtzhang, caseywang}@tencent.com

1 APPENDIX OVERVIEW

Source code is available at <https://sjtuplayer.github.io/projects/SaRA>. This appendix provides additional analysis and experiments related to SaRA, including:

- More implementation details (Sec. 2);
- More comparison results on downstream dataset fine-tuning (Sec. 3);
- More comparison results on image customization (Sec. 4);
- Comparison experiments on controllable video generation (Sec. 5);
- Scaling weight for SaRA parameter (Sec. 6);
- Merging different SaRA parameters (Sec. 7);
- More ablation studies (Sec. 8);
- Analysis on training efficiency (Sec. 9);
- Further analysis to understand what SaRA have learned (Sec. 10);
- Hyperparameter analysis (Sec. 11);
- More analysis on the learned matrix ΔP (Sec. 12);
- Limitations (Sec. 13).

2 MORE IMPLEMENTATION DETAILS

Metrics. We evaluate the generation models by three metrics: 1) **Fréchet Inception Distance (FID)** (Heusel et al., 2017) to measure the similarity between the generated image distribution and target image distribution, where a lower score indicates better similarity; 2) **CLIP Score** to measure the matching degree between the given prompts and generated images with a CLIP L/14 backbone (Radford et al., 2021), where a higher score indicates better consistency; 3) Additionally, since FID and CLIP scores exhibit a certain degree of mutual exclusivity in finetuning a text-to-image model to downstream tasks (*i.e.*, an overfitted model will result in the best FID but the worst CLIP score), we introduce a new metric, the **Visual-Linguistic Harmony Index (VLHI)**, which is calculated by adding the normalized FID and CLIP scores, to balance the evaluation of style (FID) and the preservation of model priors (CLIP score), where a higher score indicates better performance.

Visual-Linguistic Harmony Index (VLHI). We propose VLHI to evaluate both the style and the generalization of each PEFT method, by balancing FID and CLIP Score. For a group of FIDs $\{FID_i\}_{i=1}^n$ and CLIP Scores $\{CLIP_i\}_{i=1}^n$, we compute the normalized FID and CLIP Score as VLHI:

$$VLHI_i = \frac{\max(\{FID_i\}_{i=1}^n) - FID_i}{\max(\{FID_i\}_{i=1}^n) - \min(\{FID_i\}_{i=1}^n)} + \frac{CLIP_i - \min(\{CLIP_i\}_{i=1}^n)}{\max(\{CLIP_i\}_{i=1}^n) - \min(\{CLIP_i\}_{i=1}^n)} \quad (1)$$

*Equal Contribution

†Corresponding Author

For downstream dataset fine-tuning experiments, we regard the methods in one Stable Diffusion version and one dataset as a group.

Dataset details. In the downstream dataset fine-tuning experiments, we choose 5 widely-used datasets from CIVITAI with 5 different styles to conduct the fine-tuning experiments, which are Barbie Style, Cyberpunk Style, Elementfire Style, Expedition Style, and Hornify Style. Each dataset contains about 200 \sim 400 images, and for each image, we employ BLIP model (Li et al., 2022) to generate its text annotations. The detailed number of images in each dataset is recorded in Tab. 1

Dataset Image Number	Barbie	Cyberpunk	ElementFile	Expedition	Hornify
	316	440	156	396	236

Table 1: The number of images in each dataset.

Training Details. We use AdamW (Loshchilov et al., 2017) optimizer to train the methods for 5000 iterations with batch size 4, with a cosine learning rate scheduler, where the initial learning rate lr is calculated corresponds to the thresholds θ_t : $lr = 10^{-3} \times e^{-350\theta_t}$ (refer to Sec. 11). For the training images and labeled captions, we recaption them by adding a prefix ‘*name style,*’ (*name* corresponds to the dataset name) before each caption, which is a common trick in fine-tuning Stable Diffusion models to a new domain.

Implementation of SaRA. To enable easy implementation of SaRA, we have efficiently encapsulated it, allowing users to perform SaRA-based fine-tuning by modifying just a single line of training code. As shown in Algorithm 1, we integrate SaRA into the optimizer class, so users only need to replace the original PyTorch optimizer with the SaRA optimizer to automatically initiate SaRA training (The code that needs to be modified is highlighted in green.). The learning rate will be automatically assigned based on the threshold θ_t if it is not specified.

Algorithm 1 SaRA Fine-tuning Pseudocode

```

1: model = Initialize_model()
2: # optimizer = AdamW(model.parameters())
   optimizer = AdamW-SaRA(model, threshold =  $\theta_t$ )
3: for epoch = 1 to  $N$  do
4:   for each mini-batch  $(x, y)$  do
5:      $y_{pred} = \text{model}(x; \theta)$ 
6:      $loss = \text{Loss.Func}(y_{pred}, y)$ 
7:      $loss.backward()$ 
8:     optimizer.step()
9:   end for
10: end for

```

3 MORE COMPARISON RESULTS ON DOWNSSTREAM DATASET FINE-TUNING

Visualization results. We compare our model with LoRA (Hu et al., 2021), Adaptformer (Chen et al., 2022), LT-SFT (Ansell et al., 2021) and full-parameter finetuning method. We train all methods for 5,000 iterations and use the trained models to generate 500 images based on 500 text descriptions (generated by GPT-4). The quantitative results are presented in the main paper. In this section, we show more qualitative results on Stable Diffusion 1.5, 2.0, and 3.0 with resolutions 512, 768, and 1,024. The results from Stable Diffusion 1.5, 2.0, and 3.0 are shown in Figs. 1- 3 respectively. It can be seen that our model generates images that contain most of the features in the target domain and are well consistent with the given prompts under different datasets. Moreover, to show the generation diversity of Our SaRA, we further generate more images by the trained SaRA weights on Stable Diffusion 1.5, 2.0, and 3.0, where for each SaRA weight, we generate 5 images with the same prompt and different random seeds. The generated results are shown in Fig. 4- 6. It can be seen that SaRA can generate the target-domain images with high diversity, while keeping the semantics consistent with the given prompts, demonstrating a good preservation of the model prior.

More compared methods. In this section, we compare our model with additional state-of-the-art parameter fine-tuning methods on Stable Diffusion 1.5, including DoRA (Liu et al., 2024) and DiffPruning (Guo et al., 2020), which are the representative reparameterized PEFT and selective PEFT approaches, respectively. The comparison results are presented in Tab. 2. The results show that DoRA performs comparably to LoRA, while DiffPruning cannot learn enough task-specified knowledge, which results in an extremely high FID. In contrast, our model achieves the best performance as evaluated by VLHI, attaining the lowest FID and competitive CLIP score. Moreover, to demonstrate the effectiveness of our SaRA method among various selective PEFT approaches, we

Backbone	Params	Model	BarbieCore			Cyberpunk			ElementFire			Expedition			Hornify			Mean		
			FID ↓	CLIP ↑	VLHI ↑	FID ↓	CLIP ↑	VLHI ↑	FID ↓	CLIP ↑	VLHI ↑	FID ↓	CLIP ↑	VLHI ↑	FID ↓	CLIP ↑	VLHI ↑	FID ↓	CLIP ↑	VLHI ↑
SD 1.5	50M	DoRA	158.40	29.48	1.43	119.06	28.16	1.49	171.96	27.67	1.41	131.33	26.94	1.24	150.33	26.83	1.44	146.22	27.82	1.53
		LoRA	161.88	29.93	1.34	117.49	28.22	1.62	181.66	27.47	1.20	136.31	27.39	1.32	156.36	26.80	1.28	150.74	27.96	1.45
		Adaptformer	166.09	29.00	1.00	126.21	27.13	0.64	151.27	26.57	1.29	138.01	26.41	0.63	151.53	26.20	1.18	146.62	27.06	1.18
		LT-SFT	157.80	23.80	0.54	123.59	25.71	0.37	171.67	25.11	0.44	139.29	27.81	1.46	158.52	26.35	1.06	150.18	25.76	0.49
		SaRA (Ours)	148.54	28.60	1.75	121.67	27.30	1.02	132.67	26.77	1.63	131.56	27.34	1.48	140.36	25.40	1.15	134.96	27.08	1.55
	20M	DoRA	158.85	29.22	1.37	116.23	28.42	1.78	169.01	27.33	1.31	133.80	26.86	1.09	148.27	26.82	1.47	145.55	27.23	1.51
		LoRA	159.64	29.65	1.40	117.21	28.43	1.71	174.79	27.61	1.35	136.38	27.00	1.07	155.85	27.16	1.43	148.77	27.97	1.52
		Adaptformer	159.02	29.08	1.34	123.88	28.07	1.11	174.17	26.53	0.95	137.03	26.67	0.83	157.09	26.63	1.20	150.24	27.39	1.21
		LT-SFT	156.60	23.76	0.59	119.75	25.33	0.53	191.01	25.96	0.49	144.57	28.01	1.37	165.47	26.89	1.10	155.48	25.99	0.42
		SaRA (Ours)	153.68	29.33	1.63	116.69	28.24	1.69	138.64	26.63	1.50	129.98	27.04	1.36	145.62	26.40	1.39	136.92	27.53	1.69
	5M	DoRA	156.61	29.07	1.45	113.26	27.62	1.74	178.70	27.57	1.28	135.59	26.88	1.02	161.21	27.34	1.37	149.07	27.70	1.38
		LoRA	163.80	29.93	1.25	117.58	28.32	1.65	184.99	27.74	1.25	137.96	27.10	1.07	153.57	26.93	1.40	151.58	28.00	1.44
		Adaptformer	164.22	29.37	1.14	120.98	28.11	1.33	184.84	26.66	0.84	143.01	27.35	1.01	171.34	26.85	0.94	156.88	27.67	1.13
		LT-SFT	169.24	24.23	0.08	127.01	25.43	0.03	202.47	26.90	0.68	153.49	27.96	0.97	176.41	27.34	1.00	165.72	26.37	0.27
		SaRA (Ours)	153.69	29.39	1.64	118.74	28.17	1.52	174.86	27.04	1.13	134.45	27.06	1.18	157.24	26.97	1.33	147.80	27.73	1.44
	10M	DiffPruning	217.43	31.41	1.00	180.25	28.43	1.00	241.72	27.49	0.91	184.56	28.67	1.00	206.73	28.30	1.00	206.14	28.86	1.00
	860M	Full-finetune	147.81	27.77	1.65	120.22	27.84	1.47	136.49	25.10	0.95	129.07	26.75	1.21	134.86	24.64	1.00	133.69	26.42	1.30

Table 2: Comparison with different parameter-efficient fine-tuning methods (including additional DoRA and DiffPrune) on Stable Diffusion 1.5. For most of the conditions, our model achieves the best FID and VLHI score, indicating that our model learns domain-specific knowledge successfully while keeping the prior information well.

Backbone	Params	Model	BarbieCore			Cyberpunk			ElementFire			Expedition			Hornify			Mean		
			FID ↓	CLIP ↑	VLHI ↑	FID ↓	CLIP ↑	VLHI ↑	FID ↓	CLIP ↑	VLHI ↑	FID ↓	CLIP ↑	VLHI ↑	FID ↓	CLIP ↑	VLHI ↑	FID ↓	CLIP ↑	VLHI ↑
SD XL	50M	DoRA	164.42	31.76	1.77	126.45	29.20	1.76	175.78	28.23	0.74	139.84	27.60	1.12	164.53	27.29	0.90	154.20	28.82	1.06
		LoRA	168.59	31.68	1.51	132.38	28.96	1.26	134.22	27.65	1.25	130.37	27.30	1.34	154.78	27.32	1.38	144.08	28.58	1.45
		Adaptformer	171.33	30.69	1.06	135.74	28.71	0.83	139.71	27.34	0.92	135.68	27.11	0.98	151.20	26.94	1.15	146.73	28.16	1.06
		LT-SFT	165.41	30.20	1.24	131.08	28.65	1.16	140.62	27.48	0.97	126.10	26.97	1.30	150.94	27.11	1.34	142.83	28.08	1.21
		SaRA (Ours)	162.53	30.67	1.54	126.04	29.01	1.79	129.92	28.73	2.00	124.48	27.18	1.51	144.28	26.66	1.18	137.45	28.45	1.71
	20M	DoRA	165.18	31.41	1.62	124.22	28.95	1.75	177.07	28.25	0.72	138.72	27.64	1.20	163.20	27.28	0.95	153.68	28.71	1.03
		LoRA	163.46	31.58	1.27	132.38	28.96	1.26	139.89	28.02	1.31	131.63	27.52	1.43	157.04	27.32	1.27	144.88	28.68	1.46
		Adaptformer	168.54	31.25	1.38	137.61	28.99	0.87	155.14	28.12	0.94	137.73	27.56	1.19	159.13	27.38	1.24	151.63	28.66	1.10
		LT-SFT	178.51	31.44	0.88	131.72	29.01	1.34	149.82	28.01	1.03	140.51	27.91	1.30	154.82	27.16	1.21	151.08	28.71	1.16
		SaRA (Ours)	162.38	31.61	1.84	128.55	29.21	1.72	142.60	28.22	1.35	135.44	27.72	1.39	153.33	27.44	1.58	144.46	28.84	1.58
	5M	DoRA	166.21	31.19	1.49	124.68	29.09	1.81	174.47	28.05	0.66	139.24	27.37	1.00	165.32	27.26	0.83	153.98	28.52	0.94
		LoRA	169.38	30.97	1.25	126.26	29.01	1.73	151.41	27.80	0.86	138.03	27.41	1.07	157.21	27.01	0.94	148.56	28.44	1.13
		Adaptformer	178.61	30.88	0.71	138.76	29.21	0.92	160.38	27.99	0.72	144.51	27.63	0.94	161.77	26.96	0.68	156.81	28.53	0.76
		LT-SFT	174.77	31.65	1.16	129.10	29.15	1.64	165.69	28.08	0.62	147.41	27.58	0.78	165.84	27.11	0.65	156.56	28.71	0.88
		SaRA (Ours)	174.95	31.84	1.20	127.01	29.33	1.92	144.27	28.40	1.41	137.07	27.66	1.28	158.02	27.45	1.36	148.26	28.94	1.44
	Full-finetune	2085M	160.72	28.55	1.00	128.94	27.81	0.77	144.56	27.01	0.59	124.59	26.41	1.00	146.60	26.48	0.89	141.08	27.25	0.81

Table 3: Comparison with different parameter-efficient fine-tuning methods on Stable Diffusion XL. For most of the conditions, our model achieves the best FID and VLHI score, indicating that our model learns domain-specific knowledge successfully while keeping the prior information well.

compare SaRA with LT-SFT (Ansell et al., 2021), FishMask (Sung et al., 2021), DiffPruning (Guo et al., 2020), and an ablated method that fine-tunes the largest parameters on Stable Diffusion 1.5 with 50M trainable parameters. The qualitative comparison results are shown in Fig. 11. It can be observed that LT-SFT does not learn the target style well in the ElementFire and Hornify datasets. FishMask tends to generate artifacts as it tunes some effective parameters in the pretrained weights, disrupting part of the model priors. DiffPruning fails to capture task-specific information, resulting in outputs that differ significantly from the target style (despite the fact that we have tried different hyperparameters). Additionally, the ablated model that fine-tunes the largest parameters tends to overfit, similar to the full-parameter fine-tuning model. Since the most important parameters are all fine-tuned, it is prone to overfitting to the target domain, leading to generated images that do not align well with the given prompts. In contrast, our SaRA fits the five datasets well while preserving the model priors, indicating superior performance among the different selective PEFT methods.

More experiments on Stable Diffusion XL. In this section, we present additional comparison experiments on one of the most widely used stable diffusion models, Stable Diffusion XL 1.0 (Podell et al., 2023), capable of generating images at a resolution of 1024×1024 . The results, summarized in Tab. 3, demonstrate that our SaRA consistently achieves the best performance on Stable Diffusion XL 1.0, further validating the effectiveness and robustness of our approach.

More evaluation metrics. While the CLIP score (Radford et al., 2021) measures overall similarity between images and text, it may overlook finer details during evaluation. To address this, we incorporate the attribute evaluation metric (denoted as B-VQA) from T2I-CompBench++ (Huang et al., 2023), which assesses the alignment between generated images and input text prompts at a more granular level. The comparison results are presented in Tab. 4, showing that our model achieves the best or second-best B-VQA score in most cases, demonstrating its ability to preserve fine-grained details described in the input text prompts.

Backbone	Params	Model	BarbieCore			Cyberpunk			ElementFire			Expedition			Homify			Mean		
			FID ↓	B-VQA ↑	VLHI ↑	FID ↓	B-VQA ↑	VLHI ↑	FID ↓	B-VQA ↑	VLHI ↑	FID ↓	B-VQA ↑	VLHI ↑	FID ↓	B-VQA ↑	VLHI ↑	FID ↓	B-VQA ↑	VLHI ↑
SD 1.5	50M	Dora	158.40	0.37	1.16	<u>119.06</u>	0.42	0.58	171.96	0.48	0.96	131.33	0.52	<u>1.39</u>	<u>150.33</u>	0.49	1.31	<u>146.22</u>	0.46	1.20
		Lora	161.88	0.40	1.26	117.49	0.47	1.35	181.66	<u>0.51</u>	0.89	136.31	0.52	1.23	156.36	0.49	1.15	150.74	0.48	1.22
		Adaptformer	166.09	0.38	0.82	126.21	0.46	0.56	<u>151.27</u>	0.68	1.73	138.01	0.53	1.22	151.53	0.50	<u>1.37</u>	146.62	0.51	<u>1.60</u>
		LT-SFT	<u>157.80</u>	0.38	<u>1.27</u>	123.59	0.46	0.81	171.67	0.46	0.93	139.29	0.54	1.35	158.52	0.51	1.25	150.18	0.47	1.19
		SaRA (Ours)	148.54	0.40	1.84	121.67	0.47	<u>1.05</u>	132.67	0.50	<u>1.58</u>	<u>132.54</u>	<u>0.53</u>	1.48	140.36	0.51	1.73	135.15	<u>0.48</u>	1.75
	20M	Dora	158.85	0.37	1.14	116.23	0.44	0.98	<u>169.91</u>	0.48	<u>0.99</u>	<u>133.80</u>	0.52	<u>1.29</u>	<u>148.97</u>	0.49	<u>1.36</u>	<u>145.55</u>	0.46	1.25
		Lora	159.64	0.40	1.36	117.21	0.47	<u>1.35</u>	174.79	0.50	0.97	136.38	0.52	1.22	155.85	0.49	1.23	148.77	0.48	<u>1.29</u>
		Adaptformer	159.02	0.38	1.22	123.88	0.46	0.72	174.17	0.49	0.96	137.03	0.51	0.98	157.09	0.48	1.09	150.24	0.46	1.13
		LT-SFT	<u>156.60</u>	0.38	1.30	119.75	0.48	1.40	191.01	0.50	0.73	144.57	0.54	1.19	165.47	0.51	1.08	155.48	0.48	1.12
		SaRA (Ours)	153.68	0.40	1.64	<u>116.69</u>	<u>0.47</u>	1.50	138.64	0.50	1.49	129.88	0.54	1.75	<u>145.62</u>	<u>0.50</u>	1.83	136.92	0.48	1.72
	5M	Dora	156.21	0.37	1.21	<u>113.26</u>	0.44	<u>1.29</u>	128.30	0.47	0.84	135.59	0.52	1.24	161.21	0.48	1.01	148.99	0.46	1.12
		Lora	163.80	0.41	<u>1.25</u>	<u>112.58</u>	0.46	1.27	184.99	0.50	0.83	137.96	0.52	1.17	153.57	0.49	1.22	151.58	0.48	<u>1.20</u>
		Adaptformer	164.22	0.34	0.64	120.98	0.46	1.03	184.84	0.49	0.82	143.01	<u>0.53</u>	1.11	171.34	0.49	0.79	156.88	0.46	0.94
		LT-SFT	169.24	0.40	0.91	127.01	0.49	1.00	202.47	0.52	0.61	153.49	0.56	1.00	176.41	0.53	1.00	165.72	0.50	0.94
	Full-finetune	860M	147.81	0.30	1.00	120.22	0.47	1.15	136.49	0.26	0.95	129.07	0.48	1.00	134.86	0.40	1.00	133.69	0.38	1.00

Table 4: Comparison on FID and B-VQA from T2i-compbench++ (Huang et al., 2023) with different parameter-efficient fine-tuning methods on Stable Diffusion 1.5.

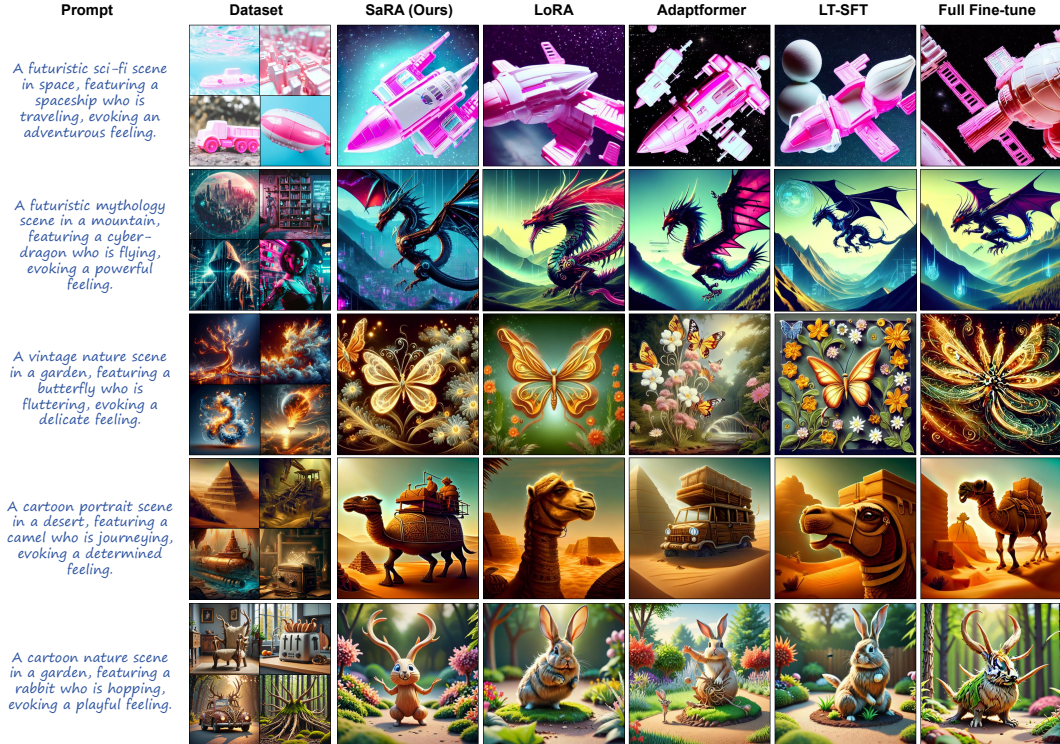


Figure 1: Comparison results between different PEFT methods on Stable Diffusion 1.5.

4 MORE COMPARISON RESULTS ON IMAGE CUSTOMIZATION

Image Customization. Image customization aims to learn a common subject from a few images and then apply it to new images. Dreambooth (Van Le et al., 2023) trains the UNet of a diffusion model to bind the target subject to a rare token and then generates images with the specified content based on the rare token. Since Dreambooth requires fine-tuning the UNet network, we compare the performance of full-finetune (original Dreambooth), LoRA, Adaptformer, LT-SFT, and our method in image customization. We compute the CLIP-Text score and CLIP-IMG Score for the generated data, along with VLHI balancing both the two metrics. As shown in Tab. 5, LoRA achieves a high CLIP-IMG score but the lowest CLIP-Text score, indicating a severe overfitting problem. Other PEFT methods, including full-parameter fine-tuning, achieve relatively low CLIP-IMG and CLIP-Text scores. In contrast, our method achieves the best CLIP-Text score, a competitive CLIP-IMG score (only lower than the overfitted LoRA), and the best average VLHI score across three datasets, demonstrating its effectiveness in image customization tasks. We further conduct the qualitative comparison on fine-tuning Dreambooth. As shown in Fig. 7, our method can learn the subject content well while preserving the prior information of the diffusion model, thereby improving the

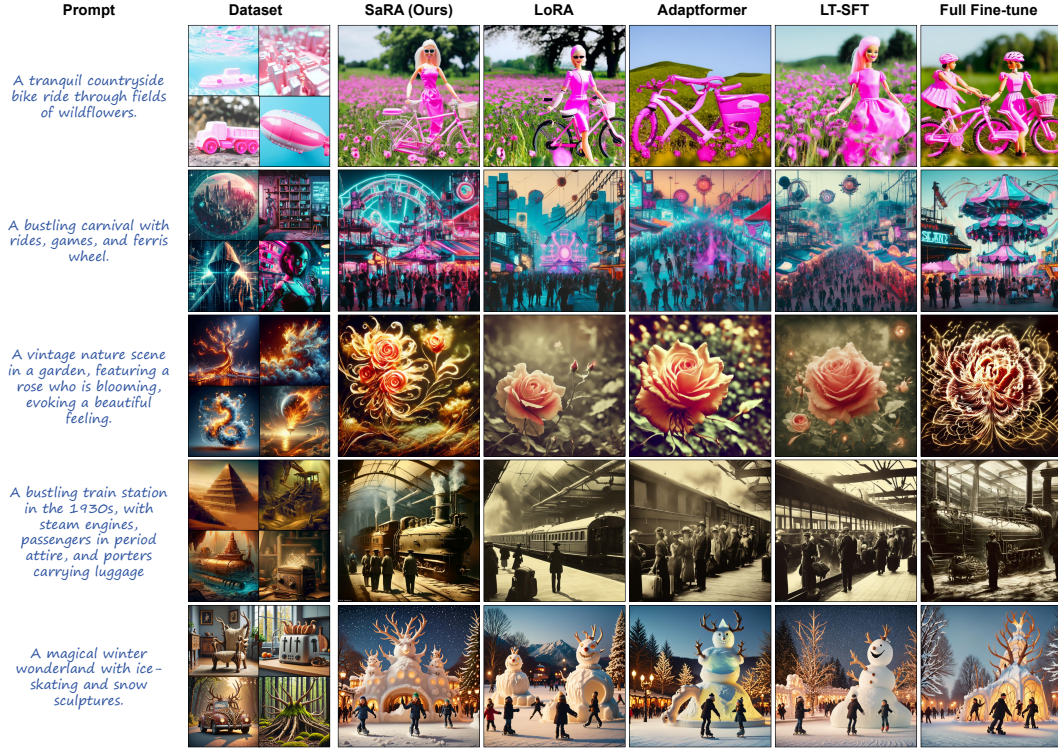


Figure 2: Comparison results between different PEFT methods on Stable Diffusion 2.0.

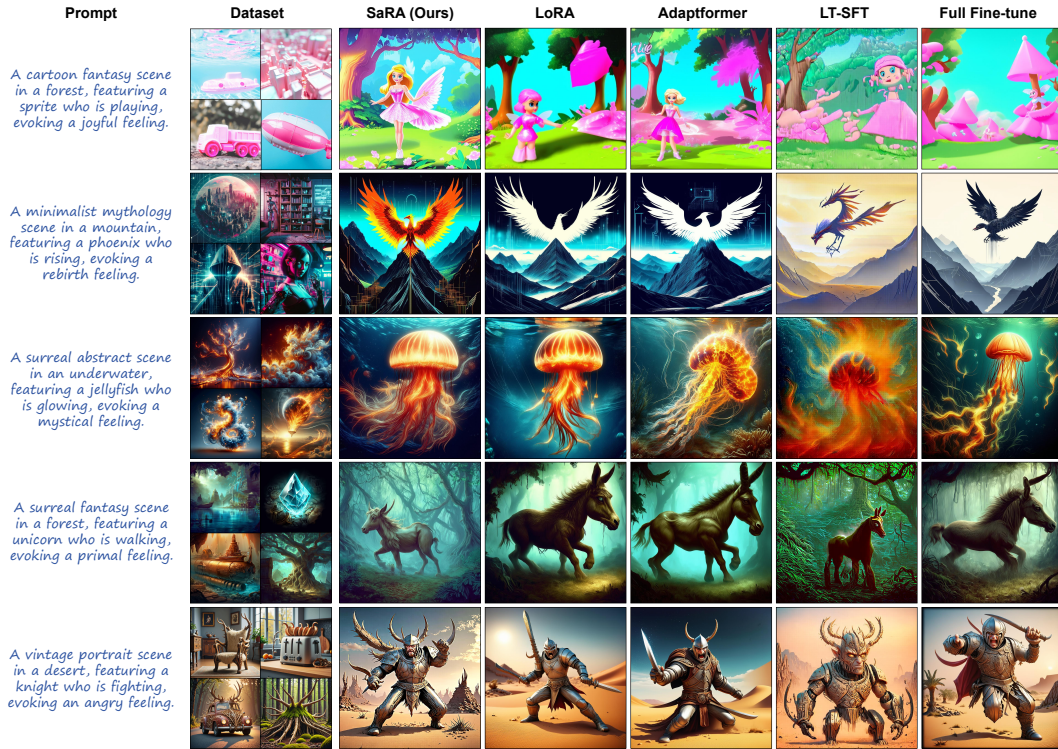


Figure 3: Comparison results between different PEFT methods on Stable Diffusion 3.0.

consistency between the generated images and the given texts, which demonstrates the effectiveness of SaRA in image customization.

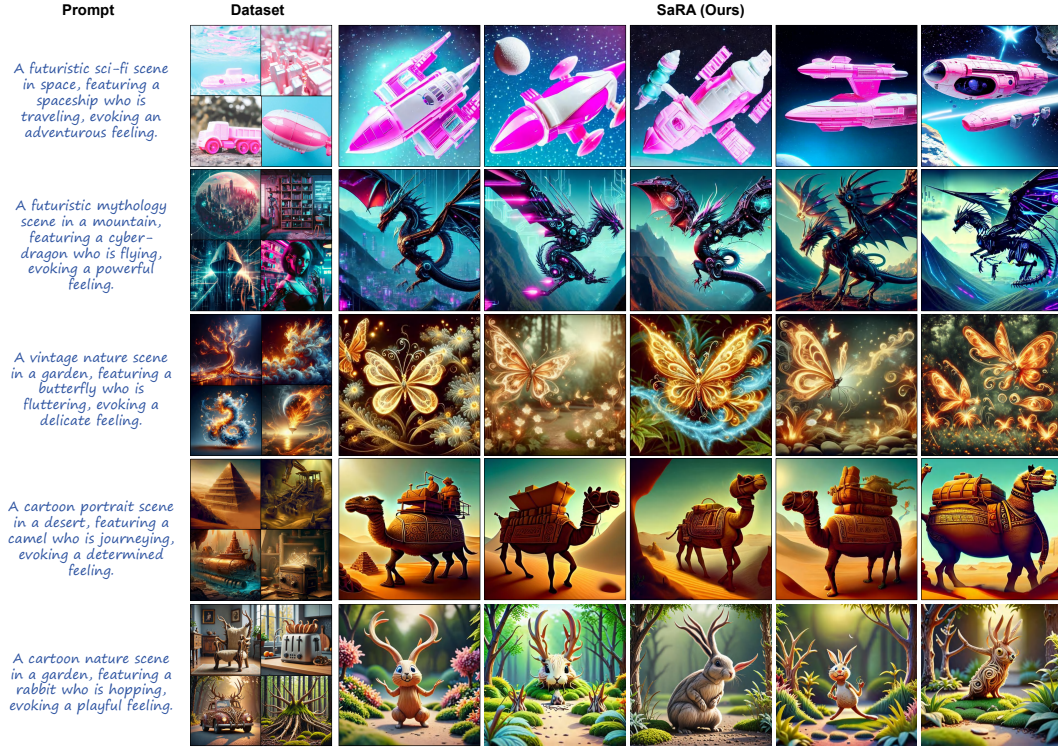


Figure 4: More generation results by SaRA for different downstream datasets on SD 1.5.

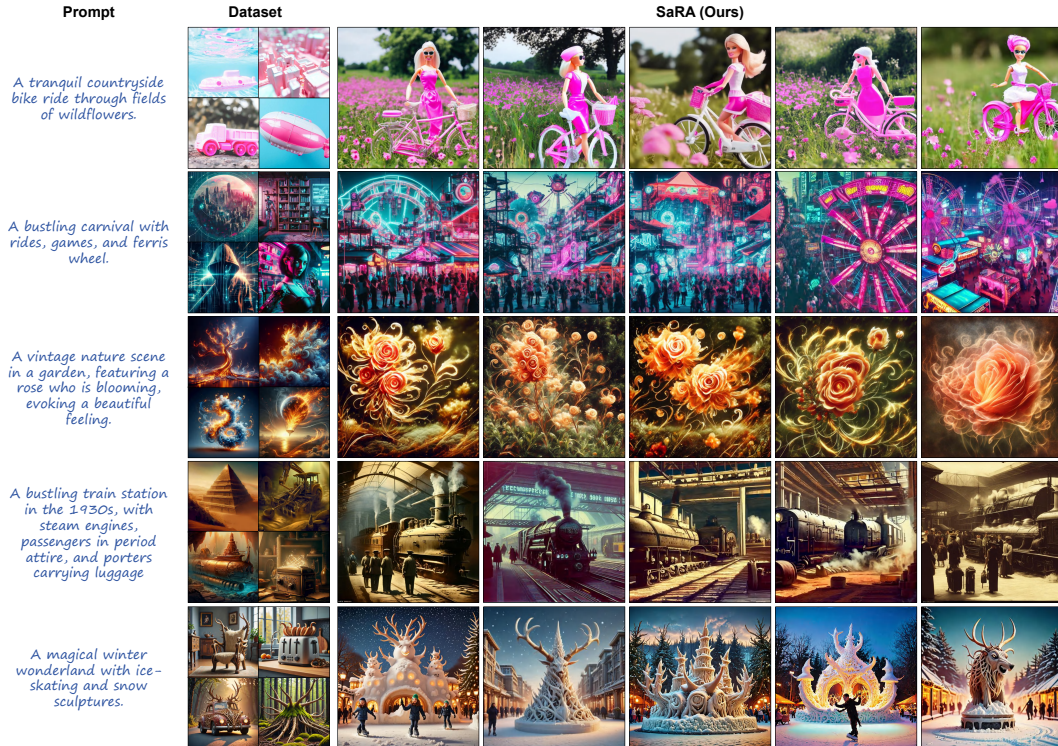


Figure 5: More generation results by SaRA for different downstream datasets on SD 2.0.

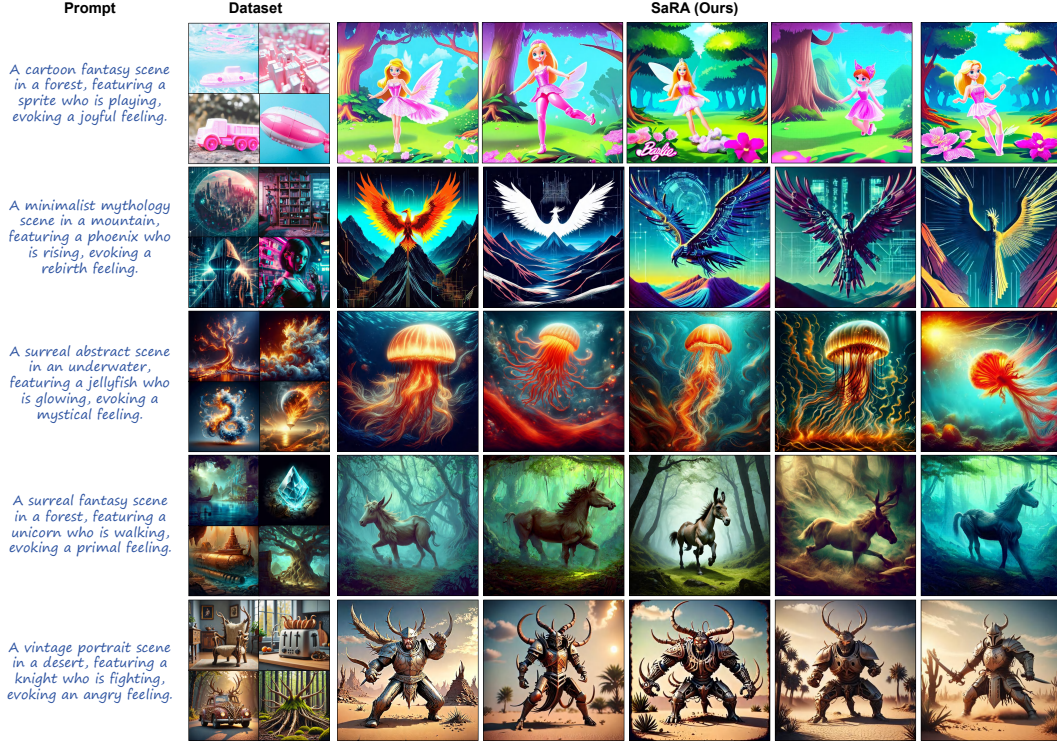


Figure 6: More generation results by SaRA for different downstream datasets on SD 3.0.

Methods	CLIP-I \uparrow	Dog CLIP-T \uparrow	VLHI \uparrow	CLIP-I \uparrow	Clock CLIP-T \uparrow	VLHI \uparrow	CLIP-I \uparrow	Backpack CLIP-T \uparrow	VLHI \uparrow	CLIP-I \uparrow	Mean CLIP-T \uparrow	VLHI \uparrow
Textual Inversion	0.788	23.94	0.36	0.789	24.15	1.00	0.654	24.09	0.00	0.744	24.06	0.39
Dreambooth + Full Fine-tune	0.776	<u>25.85</u>	<u>1.13</u>	<u>0.894</u>	22.39	1.13	0.856	25.44	<u>1.76</u>	0.842	24.56	1.36
Dreambooth + LoRA	0.895	23.64	1.00	0.913	21.71	1.00	0.917	25.23	1.84	0.908	23.53	1.00
Dreambooth + Adaptformer	0.772	25.42	0.91	0.885	23.18	<u>1.38</u>	0.873	25.25	1.69	0.843	<u>24.62</u>	<u>1.41</u>
Dreambooth + LT-DFT	0.757	23.94	0.13	0.893	22.45	1.14	0.869	25.00	1.49	0.840	23.80	0.79
Dreambooth + SaRA (Ours)	<u>0.790</u>	25.87	1.24	0.887	<u>23.51</u>	1.53	<u>0.886</u>	<u>25.27</u>	<u>1.76</u>	<u>0.854</u>	24.88	1.67

Table 5: Quantitative comparisons between different PEFT methods on image customization.

5 CONTROLLABLE VIDEO GENERATION.

We further investigate the effectiveness of our method in fine-tuning video generation models. AnimateDiff (Guo et al., 2023) is a representative video generation model based on Stable Diffusion (Rombach et al., 2022), which inserts temporal attention modules between the original spatial attention modules to model temporal correlations, enabling a diverse text-to-video generation. To achieve more controllable generation, AnimateDiff fine-tunes the temporal attention module using different camera motion data, such as Pan Left, Pan Right, Zoom In, and Zoom Out, to control the camera movements precisely. We compare the effectiveness of various PEFT methods in fine-tuning AnimateDiff for three types of camera movements, including Zoom In, Zoom Out, and Pan Right. Specifically, we collected 1,000 video-text pairs with identical camera movements for each type of camera motion. The temporal attention modules are fine-tuned using full fine-tuning, LoRA, Adaptformer, LT-SDT, and our SaRA. As shown in Fig. 8, the compared methods usually suffer from generating artifacts in the results (shown in red boxes), indicating that these methods have lost some model priors of specific content during the fine-tuning process. Moreover, for the sea turtle examples, full fine-tuning, LoRA, and LT-SFT exhibit noticeable content degradation. And for the pan-right examples, all the compared methods fail to capture the photographer, indicating a significant model overfitting problem. In contrast, our method achieves excellent camera motion control while achieving good consistency between the video content and the text.



Figure 7: Qualitative comparisons among different PEFT methods on image customization by fine-tuning the UNet model in Dreambooth (Van Le et al., 2023). Our model can accurately capture the target feature while preventing the model from overfitting, outperforming Dreambooth with other PEFT methods and Textual inversion (Gal et al., 2022).

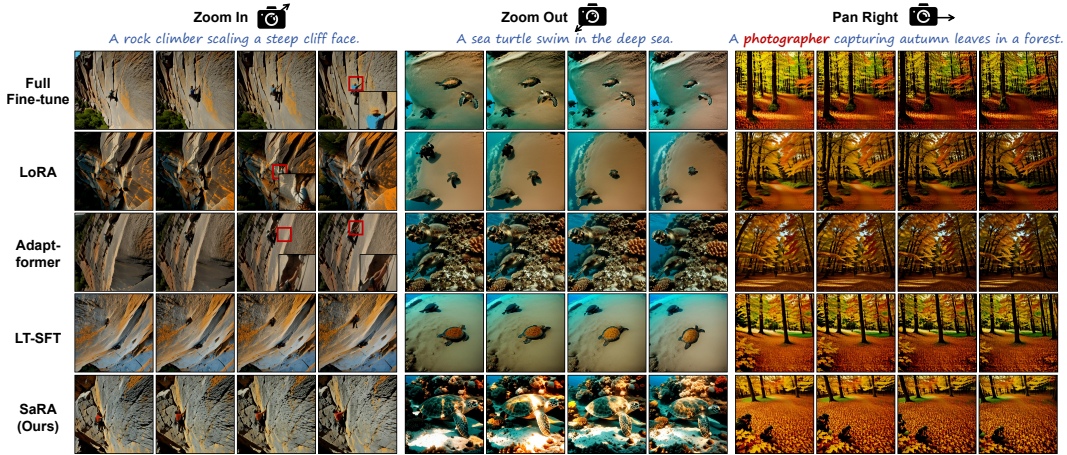


Figure 8: The comparison results of the video generation model (Guo et al., 2023), fine-tuned using different PEFT methods on three video datasets featuring zoom-in, zoom-out, and pan-right camera motions. The red boxes highlight artifacts generated by the compared methods, indicating that these methods have lost some model priors of specific content during the fine-tuning process. In the sea turtle examples, full fine-tuning, LoRA, and LT-SFT exhibit noticeable content degradation. And for the pan-right examples, all the compared methods fail to capture the photographer, indicating significant model overfitting. In contrast, our method achieves excellent camera motion control while preserving video content well.

6 SCALING WEIGHT FOR SARA PARAMETERS

Our SaRA aims to learn a sparse low-rank parameter matrix ΔP , which is added to the pre-trained weights P_0 . Similar to LoRA (Hu et al., 2021), when applying the learned parameter ΔP to the pre-trained one, we can assign a scaling weight α for the ΔP to control the emphasis extent on the learned target-domain knowledge by:

$$P = P + \alpha \Delta P. \quad (2)$$

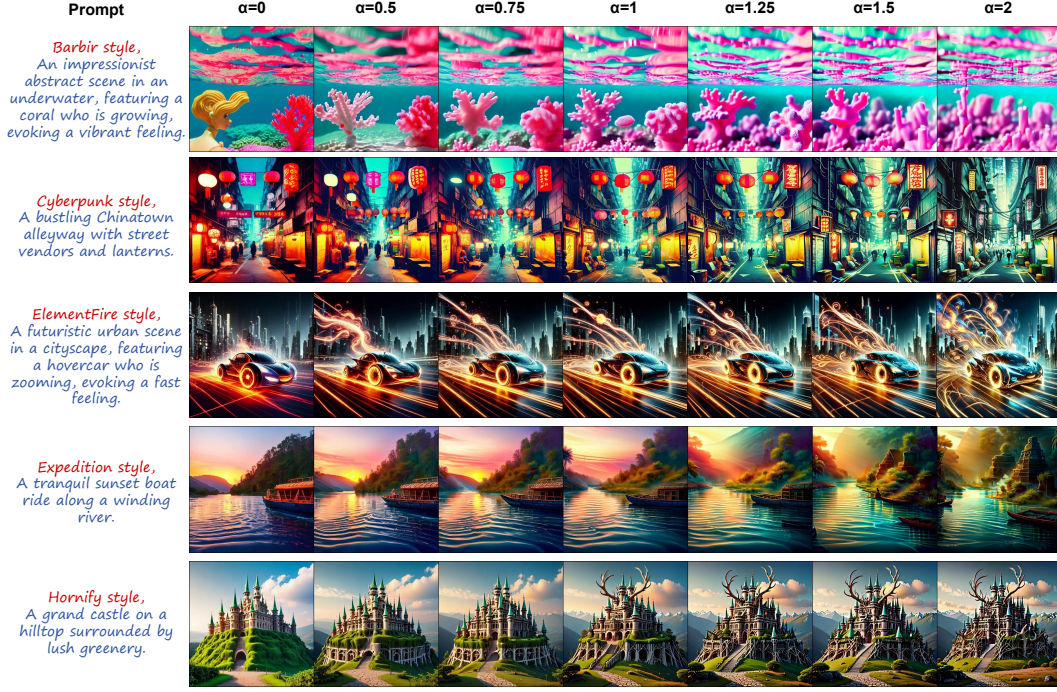


Figure 9: The generated results from different weight α for the learned SaRA parameters by Stable Diffusion 1.5. As α increases, the generated image contains more tar-get domain features.

We show the results on different α ranging from 0 to 2 on five datasets in Fig. 9. It can be seen that as the scaling weight α increases, the model tends to generate images with more target-domain features, but may lose part of the information specified by the given texts.

7 MERGING DIFFERENT SARA PARAMETERS

For two SaRA parameters ΔP_1 and ΔP_2 learned from two different datasets, we aim to find whether they can be combined to form new parameters that contain the knowledge from both the two datasets. We combine the two parameters by:

$$\Delta P = \alpha_1 \Delta P_1 + \alpha_2 \Delta P_2. \quad (3)$$

Then, we employ the combined SaRA parameter ΔP to generate images. We choose four combinations: 'Barbie Style' + 'Cyberpunk style', 'Cyberpunk Style' + 'ElementFire style', 'ElementFire Style' + 'Expedition style', and 'Hornify Style' + 'Cyberpunk style', where we simply assign $\alpha_1 = \alpha_2 = 0.6$. The generated images are shown in Fig. 10. It can be seen that, after combining the SaRA parameters learned from two different datasets, the output images contain the features from both two datasets, which indicates that we can merge different SaRA parameters together, enabling more flexible and abundant generation results.

8 MORE ABLATION STUDIES.

Ablation study on threshold. In SaRA, the threshold is an important hyperparameters that influence the size of the parameter space directly. We conduct experiments on different thresholds ranging from $2e-4$ to $1e-1$ on the Expedition dataset and Stable Diffusion 1.5, and evaluate the generated images by FID and CLIP score. The results are shown in Tab. 6. It can be seen that when the threshold is too small (e.g., $2e-4$), the FID becomes much higher, indicating learning less target

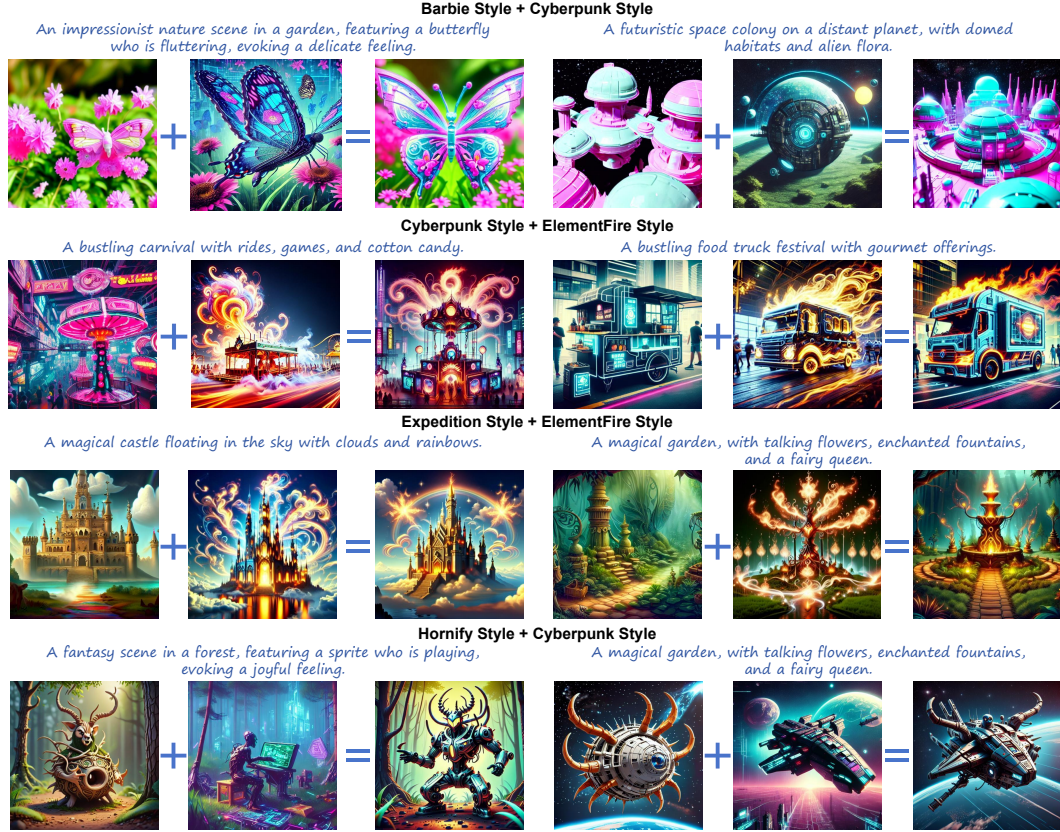


Figure 10: Combining the SaRA parameters learned from two different datasets, the model can generate images with features from both datasets. We show the combination results for 'Barbie Style' + 'Cyberpunk style', 'Cyberpunk Style' + 'ElementFire style', 'ElementFire Style' + 'Expedition style', and 'Hornify Style' + 'Cyberpunk style' in this figure.

domain knowledge. And when the threshold is large (i.e., $\text{threshold} \leq 2e-3$), the model performs quite stably. Since we have a low-rank loss, the model with a high threshold also keeps the CLIP score well. In summary, our SaRA performs well in different thresholds, demonstrating the robustness of our model.

Threshold	2e-4	8e-4	2e-3	5e-3	1e-2	5e-2	1e-1
FID ↓	134.45	129.98	132.54	131.05	130.42	130.71	129.88
CLIP ↑	27.06	27.04	27.38	27.21	27.15	27.04	27.02

Table 6: Ablation study on the threshold.

More ablation study on the low-rank loss. In this section, we conduct more ablation studies on the low-rank loss \mathcal{L}_{rank} , where we choose Stable Diffusion 1.5 and two additional datasets (Cyberpunk and ElementFire) for the experiment. The results are shown in Tab. 7, where we can see that the mode without \mathcal{L}_{rank} always tends to get a worse CLIP score, indicating a significant performance drop. Therefore, the low-rank loss is quite necessary in our model to keep the model prior.

9 ANALYSIS ON TRAINING EFFICIENCY

In Sec.4.4 of the main paper, we propose unstructural backpropagation, which allows selective PEFT to store and update only the gradients of trainable parameters, significantly reducing memory usage

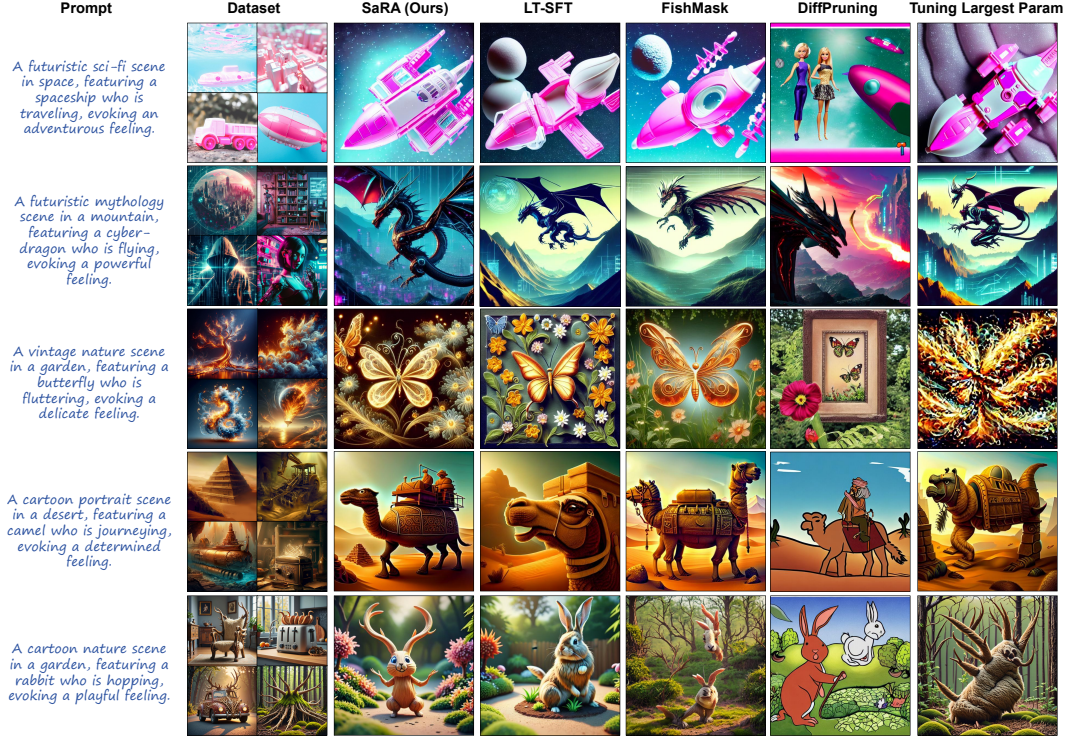


Figure 11: More qualitative comparison with the existing Selective PEFT methods (LT-SFT (Ansell et al., 2021), FishMask (Sung et al., 2021), DiffPruning (Guo et al., 2020)), and the ablated model that fine-tunes the largest parameters) on SD 1.5.

method	Cyberpunk		ElementFire	
	FID	CLIP	FID	CLIP
SaRA	121.67	27.30	132.67	26.77
SaRA w/o. \mathcal{L}_{rank}	120.33	26.52	131.56	25.88

Table 7: More ablation studies on the low-rank loss \mathcal{L}_{rank} .

during training. We conducted experiments on the Stable Diffusion 2.0 model using an 80G NVIDIA A100 GPU, comparing the memory usage and training time of LT-SFT (Selective PEFT method), LoRA, and our method across different batch sizes. The results, shown in Fig. 5 of the main paper, demonstrate that our method achieves the lowest memory consumption and training time under all batch sizes. Compared to LT-SFT, we reduce memory usage by a fixed 9.2G (equivalent to the total gradient size of fixed parameters) and achieve over 45% memory reduction for smaller batch sizes. Furthermore, compared to LoRA, our method saves over 52% memory and 49% training time for larger batch sizes, showcasing the efficiency of our SaRA in model fine-tuning.

10 FURTHER ANALYSIS TO UNDERSTAND WHAT SARA HAVE LEARNED

The Correlation between ΔP and P . We further investigate what exactly is learned by the sparse parameter matrix ΔP_{Ours} obtained through our method. Firstly, we examine the relationship between ΔP and the pre-trained parameter matrix P . We want to know whether ΔP has learned new knowledge that is not present in P , or it amplifies some existing but previously not emphasized knowledge in P . To answer this question, we study the subspaces of ΔP and P . We first conduct SVD decomposition on ΔP_{Ours} , and obtain the left and right singular-vector matrices $U_{\Delta P_{Ours}}$ and $V_{\Delta P_{Ours}}^T$. We then project P into the first r -dimensional subspace of ΔP using $U_{\Delta P_{Ours}} P V_{\Delta P_{Ours}}^T$. We quantify the correlation between P and the first r -dimensional subspace of ΔP by calculating

Rank Matrices	r=4			r=16			r=64		
	ΔP_{Ours}	ΔP_{LoRA}	P	ΔP_{Ours}	ΔP_{LoRA}	P	ΔP_{Ours}	ΔP_{LoRA}	P
$\ UPV^T\ _F$	0.17	0.34	9.36	0.48	1.14	14.82	2.68	3.90	23.91
Amplification	25.72	13.45	-	6.50	4.05	-	1.64	1.18	-

Table 8: The correlation between the learned parameter matrices ΔP and the pretrained weights P . Our learned parameter matrix ΔP_{Ours} amplifies the directions that are not emphasized in the pretrained weights P , and has a larger amplification factor than LoRA, indicating our model learns more task-specific knowledge than LoRA.

the Frobenius norm of this projection $\|U_{\Delta P_{Ours}} P V_{\Delta P_{Ours}}^T\|_F$, where a smaller norm indicates lower correlation between the subspace of ΔP and P .

For a valid reference, we further decompose the parameter matrix ΔP_{LoRA} learned by LoRA using SVD to obtain the respective $U_{\Delta P_{LoRA}}$ and $V_{\Delta P_{LoRA}}^T$ matrices, and project the pre-trained parameter matrix P into the first r -dimensional subspace of ΔP_{LoRA} using $U_{\Delta P_{LoRA}} P V_{\Delta P_{LoRA}}^T$.

In addition, we calculate an amplification factor to determine how much the parameter matrix ΔP amplifies the directions that are not emphasized by P . The amplification factor is computed as $f_a = \frac{\|\Delta P\|_F}{\|UPV^T\|_F}$. The higher the amplification factor is, the more task-specific knowledge is learned.

We investigate the relationship between the first $r = 4, 16, 64$ dimensional subspaces of ΔP and P . The results are shown in Tab. 8¹, from which we can draw the following conclusions:

1. The learned sparse matrix ΔP_{Ours} from our model has a significant amplification factor, such as **25.72** times for $r = 4$, which indicates the correlation between the first 4-dimensional subspace of ΔP_{Ours} and P is low, and ΔP_{Ours} primarily amplifies the directions that are not emphasized in P .
2. Compared to the low-rank parameter matrix ΔP_{LoRA} learned by LoRA, our model achieves a higher amplification factor across different values of r , indicating that our method can learn more knowledge that is not emphasized in P than LoRA.
3. As r increases, the amplification factor gradually decreases, suggesting that the knowledge learned by ΔP is mostly contained within P , and the primary role of ΔP is to amplify some of the existing but previously not emphasized knowledge in P (if new knowledge that is not present in P has been learned by ΔP , the correlation should remain low as r increases, and the amplification factor should remain high, which is not the case presented in our experiments).

The Correlation between $P + \Delta P$ and P . We aim to further understand whether our learned parameter matrix ΔP_{Ours} disrupts the information in the original parameter space (spanned by the pretrained weights P), which may lead to overfitting and loss of prior information. To analyze the preservation of prior information, we calculate the correlation between the final updated parameter matrix ($P + \Delta P$) and the pretrained weights P . Specifically, we calculate the similarity between the subspaces of $(P + \Delta P)$ and P . We decompose $(P + \Delta P)$ and P using Singular Value Decomposition (SVD) to obtain the left-singular unitary matrices U , and examine the similarity between the subspaces spanned by the first r_i singular vectors of $U_{P+\Delta P}$ and the first r_j singular vectors of U_P . We quantify the subspace similarity using the normalized subspace similarity based on the Grassmann distance (Hu et al., 2021):

$$\phi(P_1, P_2, r_i, r_j) = \frac{\|U_1^{r_i T} U_2^{r_j}\|_F^2}{\min(r_i, r_j)} \in [0, 1], \quad (4)$$

where $U_k \Sigma_k V_k^T = SVD(P_k), k = \{1, 2\}$.

We calculate the similarity between the pre-trained parameter matrix P and the updated parameter matrices obtained from three approaches: 1) our model ($P + \Delta P_{Ours}$), 2) LoRA ($P + \Delta P_{LoRA}$), and 3) a random parameter matrix added to the pre-trained parameters ($P + \Delta P_{Random}$). The results are shown in Fig. 12. As a reference, the subspace similarity $\phi(P + \Delta P_{Random}, P, r_i, r_j)$ of randomly updated parameters approaches zero across different dimensions r_j and r_i , indicating ran-

¹ $\|\Delta P_{Ours}\|_F = 4.40$ and $\|\Delta P_{LoRA}\|_F = 4.62$.

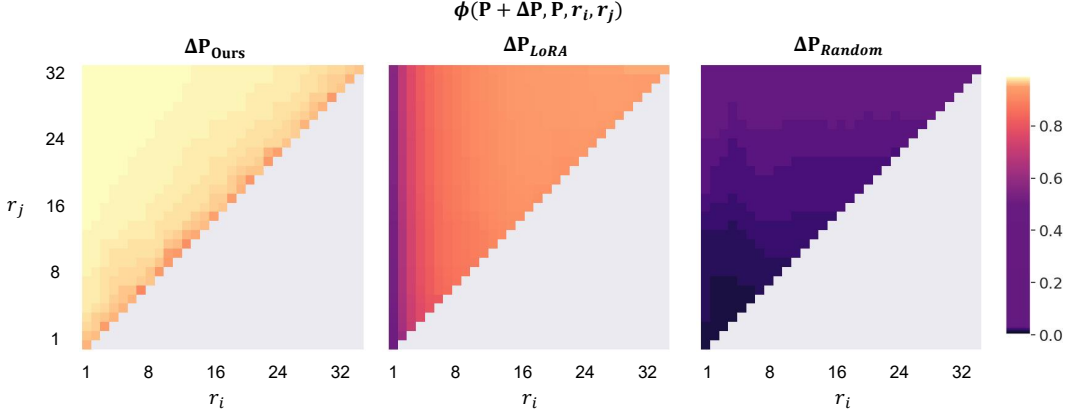


Figure 12: Subspace similarity between $P + \Delta P$ and P . The similarity $\phi(P + \Delta P_{Ours}, P, r_i, r_j)$ between our updated parameter matrix and the pretrained parameter matrix achieves a similarity larger than **96%** across different dimensions of the subspace (r_i, r_j) . The results indicate the learned sparse parameter matrix ΔP of our model keeps the prior information in the pretrained parameters P well.

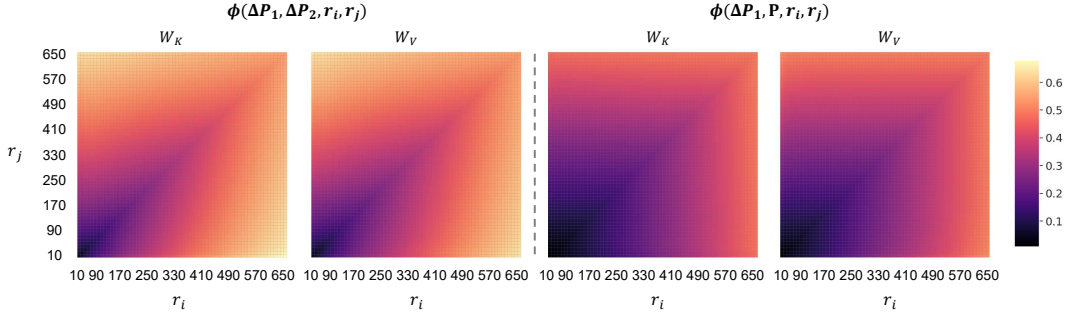


Figure 13: Subspace similarity between ΔP_1 and ΔP_2 under thresholds $2e - 3$ and $8e - 4$. The total similarity between their learned matrices exceeds 60% (when r is around 650), compared to only 40% similarity between ΔP_1 and the pre-trained weights P , demonstrating the learned matrices from different thresholds learn similar task-specific knowledge, while emphasizing different directions (relative smaller similarity when r is small).

dom weights will destroy the prior information in the pre-trained weights absolutely. In contrast, the similarity $\phi(P + \Delta P_{Ours}, P, r_i, r_j)$ between our learned parameter matrix added to the pre-trained parameters and the original parameter matrix exceeds 96% across different subspace dimensions (r_i, r_j) , indicating that our learned parameter matrix effectively preserves the information in the original parameter matrix. In addition, compared to the parameter matrix $(P + \Delta P_{LoRA})$ updated by LoRA, our updated parameter matrix $(P + \Delta P_{Ours})$ shows greater subspace similarity with the pre-trained parameters P , demonstrating that our model’s learned sparse parameter matrix better preserves the prior information of the pre-trained parameters, effectively avoiding model overfitting. Combining this with the conclusions from the previous section, we can further conclude that ***Our model can learn more task-specific knowledge, while more effectively preserving the prior information of the pre-trained parameter matrix than LoRA.***

The Correlation between ΔP under Different Thresholds. We further investigate the relationship between the learned parameter matrices ΔP under different thresholds. Our experiments focus on the matrices for Key W_K and Value W_V from the medium block’s attention modules in SD1.5. In this experiments, we select two thresholds, $2e - 3$ and $8e - 4$ (corresponding to ΔP_1 and ΔP_2), and compute the similarity of their subspaces using Eq. (4). For comparison, we also calculate the similarity between the parameter matrix ΔP_1 learned with a threshold of $2e - 3$ and the pre-trained parameter matrix P . The results are shown in Fig. 13. It can be observed that the overall similarity between the parameter matrices learned under the two thresholds exceeds 60% (peaking at

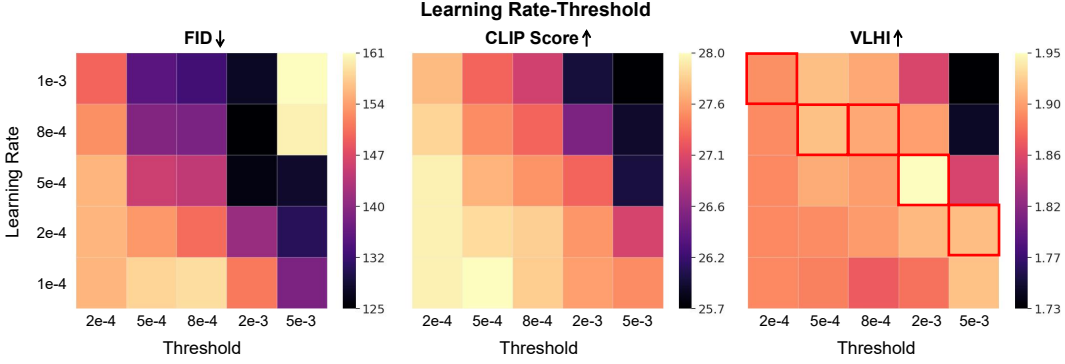


Figure 14: The comparison results on different learning rates and thresholds. The model with a larger threshold should employ a larger learning rate to learn the target-domain information well (red boxes indicate the best results).

around $r = 650$), indicating that the knowledge learned under different thresholds is roughly similar, but with different emphases (lower similarity at smaller r). In contrast, the similarity between ΔP_1 and the pre-trained parameters P is consistently below 40%. The higher similarity between ΔP_1 and ΔP_2 suggests that the learned parameter matrices from different thresholds indeed capture similar task-specific knowledge, which supports the feasibility of fine-tuning the model with fewer parameters.

11 HYPERPARAMETER ANALYSIS

In this section, we conduct experiments on different hyperparameters in our model: learning rate, progressive iteration (the iteration for progressive parameter adjustment), and the weight for rank loss λ_{rank} . We chose Stable Diffusion 1.5 and Expedition dataset for the following experiments (if not specified, the threshold $\theta_t = 2e - 3$) and evaluated the results by FID, CLIP Score, and VLHI.

Learning Rates and Thresholds. We first investigate the two most critical hyperparameters: learning rate and threshold. We selected learning rates $\{1e-4, 2e-4, 5e-4, 8e-4, 1e-3\}$ and thresholds $\{2e-4, 5e-4, 8e-4, 2e-3, 5e-3\}$ for our experiments, resulting in a total of 25 models. The quantitative results are shown in Fig. 14. It can be observed that, for the same learning rate, as the threshold increases, the model’s FID gradually decreases while the CLIP Score gradually increases. This indicates that a larger learnable parameter set can learn more task-specific information, but is also more likely to lose pre-trained prior knowledge. For the same threshold, increasing the learning rate yields similar results. However, for relatively large thresholds (e.g., $5e - 3$), a high learning rate (e.g., $8e - 4$ and $1e - 3$ in the figure) may cause the model training to collapse. Therefore, selecting an appropriate learning rate is crucial for achieving good results.

We further use the VLHI metric to analyze the performance of the models trained with different learning rates under various thresholds by balancing FID and CLIP scores, as shown in the third column in Fig. 14. The optimal learning rate for each threshold is marked with a red box. It can be seen that as the threshold increases, a gradually decreasing learning rate should be used to prevent severe overfitting. Conversely, as the threshold decreases, a larger learning rate should be employed to enhance the model’s ability to learn task-specific knowledge. In summary, there is a negative correlation between the learning rate and the threshold. To adaptively select an optimal learning rate, we fit an exponential function $f(x) = a \times e^b$ using the five data points shown in the figure.

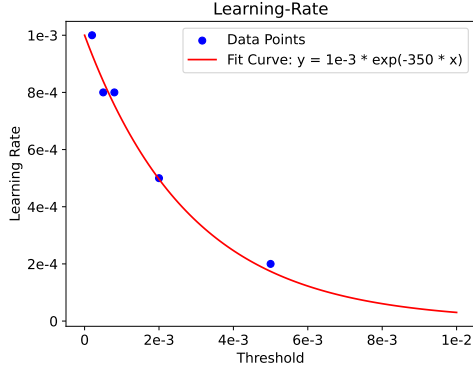


Figure 15: The fit curve of the best pairs of learning rate and threshold.

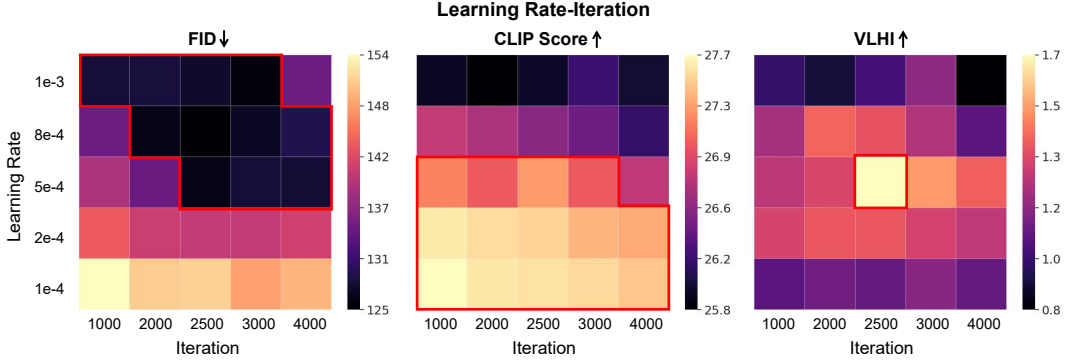


Figure 16: The comparison results on different learning rates and progressive iterations. A larger learning rate or progressive iteration improves the FID while sacrificing the CLIP Score.

The resulting function for adaptively computing the learning rate for different thresholds is shown in Fig. 15. The curve fits the five data points well, and when the threshold approaches 0, the learning rate is approximately $1e-3$, which does not result in an excessively high learning rate. Similarly, for larger thresholds (e.g., $1e-2$), the learning rate is around $3e-5$, comparable to the learning rate used in full fine-tuning, avoiding an excessively low learning rate. We do not consider even larger thresholds, as these parameters are highly effective in the model, and fine-tuning them would contradict the purpose of our method. Therefore, we derive a function to adaptively compute a good learning rate Lr based on the threshold θ_t :

$$Lr = 10^{-3} \times e^{-3500\theta_t}. \quad (5)$$

Learning Rates and Progressive Iteration. We then study the effects of learning rate and progressive iteration (the iteration for progressive parameter adjustment) together. We train the models with learning rates $\{1e-4, 2e-4, 5e-4, 8e-4, 1e-3\}$ and progressive iteration $\{1000, 2000, 2500, 3000, 4000\}$, which forms 25 models in total. The quantitative results are shown in Fig. 16. For all the metrics (FID, CLIP Score, and VLHI), the brighter the color is, the better the model performs. It can be seen that, as the learning rate or progressive iteration grows, the model learns more task-specific knowledge (a better FID), while the CLIP score becomes worse. Therefore, we should balance both the learning rate and progressive iterations, where the model with learning rate $5e-4$ and progressive iteration 2000 achieves the best VLHI, reaching both a good FID and CLIP Score.

λ_{rank} and Progressive Iteration. We then analyze the influence of the weight for rank loss λ_{rank} and progressive iteration at the same time. We train the models with $\lambda_{rank} \{1e-4, 5e-4, 1e-3, 5e-3, 1e-2\}$ and progressive iteration $\{1000, 2000, 2500, 3000, 4000\}$, which constitutes 25 models in total. The quantitative results are shown in Fig. 17. It can be seen that as λ_{rank} increases, the FID becomes worse while the CLIP Score performs better, demonstrating that λ_{rank} helps the model keep the prior information in the pre-trained weights, but with a less effect in fitting the target domain. Therefore, to simultaneously reach a relatively good FID and CLIP Score, we choose $\lambda_{rank} = 5e-3$ with progressive iteration 2500, which results in the best VLHI.

12 MORE ANALYSIS ON THE LEARNED WEIGHT MATRIX ΔP

The Correlation between ΔP and P under Different Thresholds. We compute the subspace similarity between the learned matrices ΔP under different thresholds and the pre-trained weights P by Eq. (6) of the main paper. The results are shown in Fig. 18. It demonstrates that ΔP does not contain the top singular directions of W , since the overall similarity between the singular directions in the learned matrices ΔP and the top 32 directions of P is barely around 4%. And it further validates that the matrices ΔP contain more task-specific information rather than repeating the directions that are already emphasized in the pre-trained weights. Moreover, by comparing the ΔP from different thresholds, we can find that as the thresholds grow, the subspace similarity between ΔP and P becomes smaller, indicating that a larger threshold can learn more task-specific

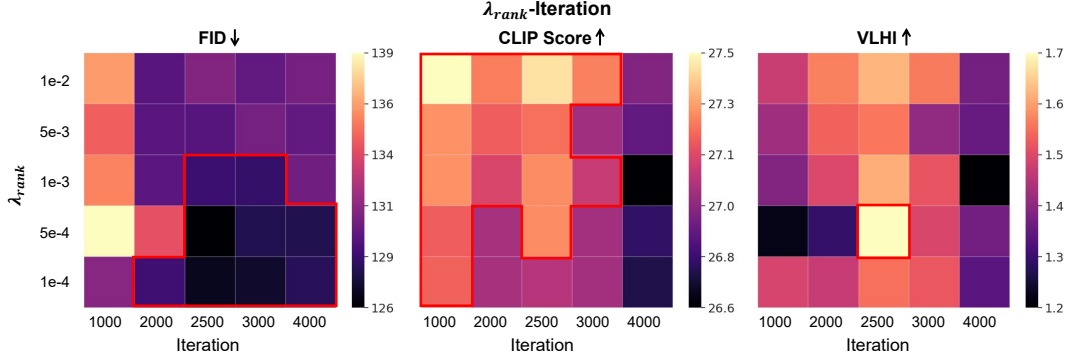


Figure 17: The comparison results on different weights λ_{rank} for rank loss and progressive iterations. A larger λ_{rank} improves the CLIP Score while sacrificing the FID.

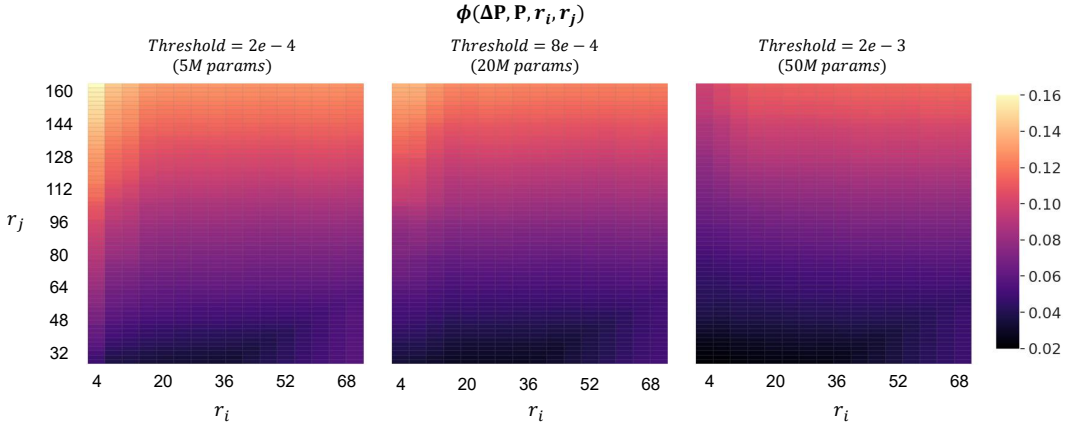


Figure 18: Subspace similarity between ΔP and P under different thresholds.

information, therefore a large threshold can contribute to a better FID as shown in Tab. 1 of the main paper.

More Analysis on ΔP from Different Layers. In the main paper, we have analyzed the subspace similarity between the learned matrices ΔP from different thresholds, and concluded that the matrices from different thresholds learn similar task-specific knowledge, but emphasize different directions. To further validate this conclusion, we conduct more quantitative analysis between different thresholds ($\theta_t = 2e - 3$ and $8e - 4$) from the attention layers in the bottom, medium, and up blocks. Moreover, we take all the learnable matrices in the attention module into consideration, including the Query, Key, Value, and FFN matrices (corresponds to W_Q , W_K , W_V , and W_{Out} respectively). The results can be referred to in Fig. 19 the x-axis and y-axis represent $\theta_t = 2e - 3$ and $\theta_t = 8e - 4$ respectively., where the heatmaps show almost the same color and distributions, indicating that our conclusion is consistent for the learned matrices from different modules and different attention layers.

Further Analysis of ΔP across Different Threshold Pairs. In the main paper, we analyzed the subspace similarity between the learned matrices ΔP from thresholds of $2e - 3$ and $8e - 4$, concluding that the learned matrices from different thresholds capture similar task-specific knowledge while emphasizing different directions. In this section, we extend our quantitative analysis to additional threshold pairs: $(2e - 3, 8e - 4)$, $(2e - 3, 2e - 4)$, and $(8e - 4, 2e - 4)$. We consider all learnable matrices in the attention module, including the Query, Key, Value, and FFN matrices (corresponding to W_Q , W_K , W_V , and W_{Out} , respectively). The results are presented in Fig. 20. The subspace similarity between the thresholds $(2e - 3, 2e - 4)$ is lower than that of $(2e - 3, 8e - 4)$ and $(8e - 4, 2e - 4)$, suggesting that matrices learned from a closer threshold pair exhibits greater subspace similarity and acquire more similar knowledge.

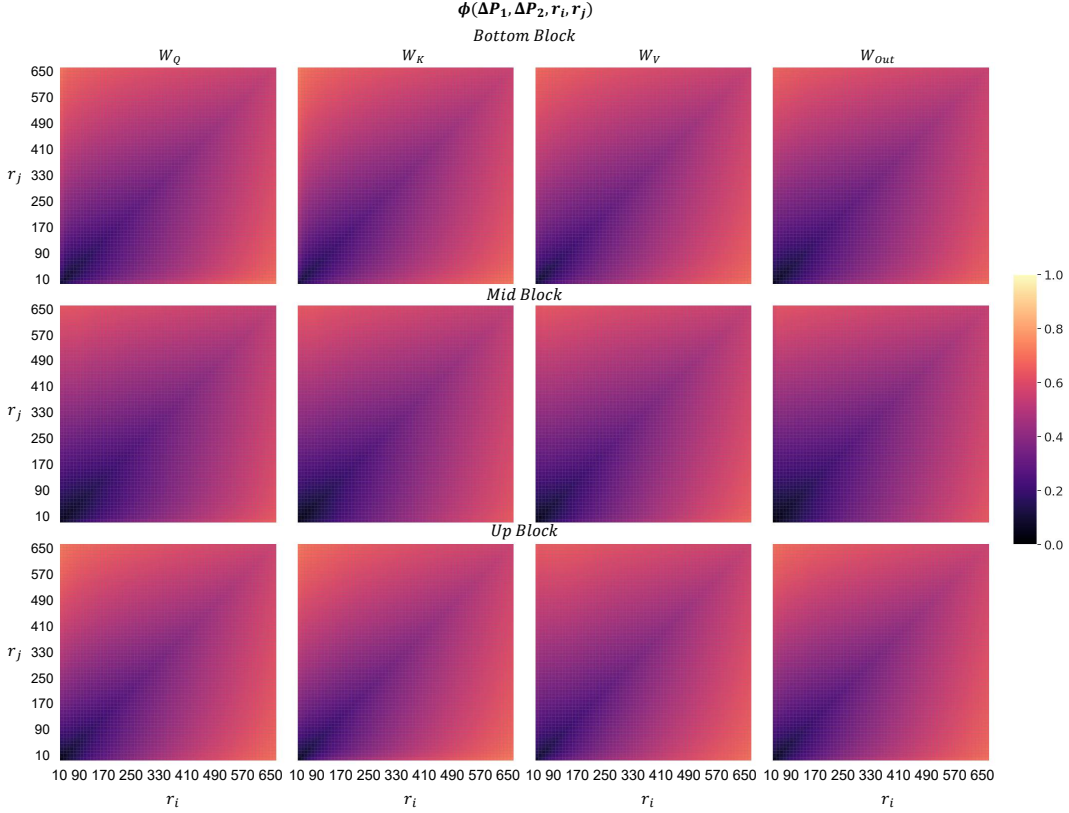


Figure 19: Subspace similarity between ΔP from threshold $\theta_t = 2e - 3$ (50M parameters) and $\theta_t = 8e - 4$ (20M parameters), in different attention layers, where the x-axis and y-axis represent $\theta_t = 2e - 3$ and $\theta_t = 8e - 4$ respectively. The subspace similarity across different layers exhibits consistent behavior, demonstrating that the knowledge learned by ΔP remains invariant across layers and modules, indicating strong robustness.

13 LIMITATIONS

Our SaRA focuses on fine-tuning the ineffective parameters of a pre-trained model. However, if the model size is relatively small (e.g., not as large as diffusion models, which typically exceed 100M parameters), the number of ineffective parameters may be insufficient to effectively adapt the model to the downstream dataset. As a result, SaRA is better suited for fine-tuning large models rather than smaller ones. Additionally, since there is no rigorous proof that parameters with the smallest absolute values are always ineffective, caution is warranted to account for potential exceptions, which could lead to reduced performance of SaRA in certain scenarios.

REFERENCES

- Alan Ansell, Edoardo Maria Ponti, Anna Korhonen, and Ivan Vulić. Composable sparse fine-tuning for cross-lingual transfer. *arXiv preprint arXiv:2110.07560*, 2021.
- Shoufa Chen, Chongjian Ge, Zhan Tong, Jiangliu Wang, Yibing Song, Jue Wang, and Ping Luo. Adaptformer: Adapting vision transformers for scalable visual recognition. *Advances in Neural Information Processing Systems*, 35:16664–16678, 2022.
- Rinon Gal, Yuval Alaluf, Yuval Atzmon, Or Patashnik, Amit H Bermano, Gal Chechik, and Daniel Cohen-Or. An image is worth one word: Personalizing text-to-image generation using textual inversion. *arXiv preprint arXiv:2208.01618*, 2022.
- Demi Guo, Alexander M Rush, and Yoon Kim. Parameter-efficient transfer learning with diff pruning. *arXiv preprint arXiv:2012.07463*, 2020.

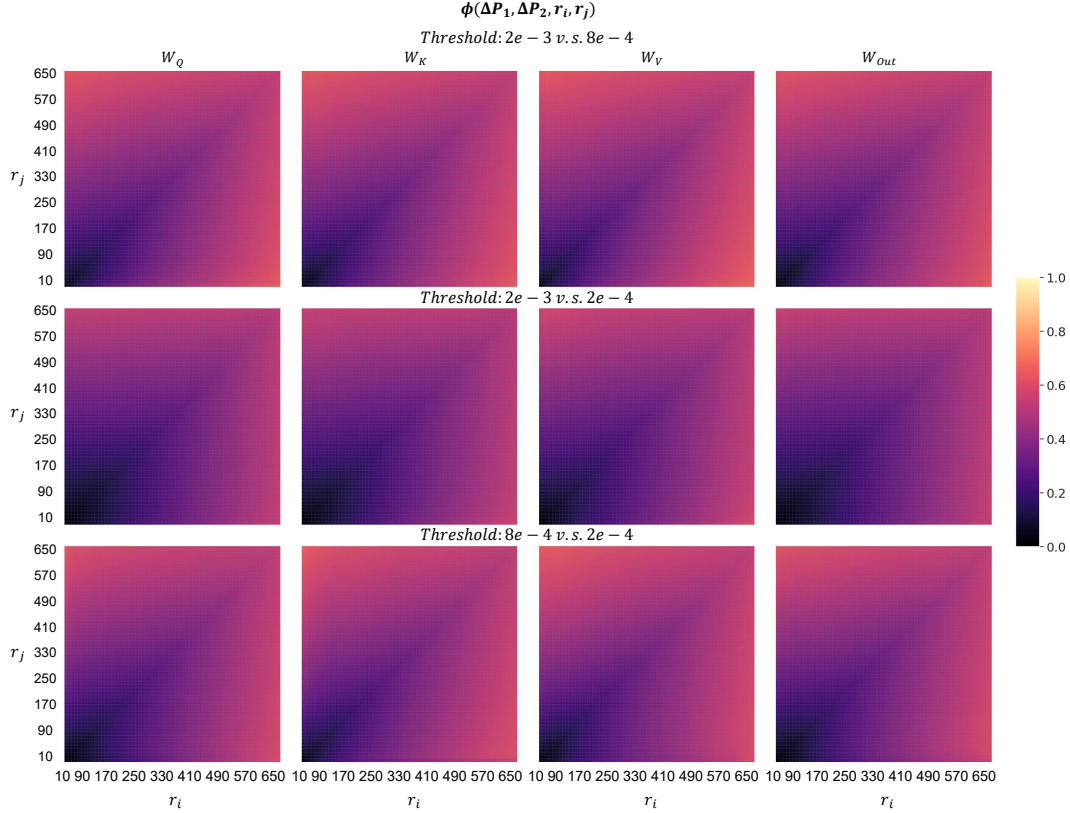


Figure 20: Subspace similarity between ΔP across different threshold pairs: $(2e-3, 8e-4)$, $(2e-3, 2e-4)$, and $(8e-4, 2e-4)$ (from top to bottom).

Yuwei Guo, Ceyuan Yang, Anyi Rao, Yaohui Wang, Yu Qiao, Dahua Lin, and Bo Dai. Animatediff: Animate your personalized text-to-image diffusion models without specific tuning. *arXiv preprint arXiv:2307.04725*, 2023.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Kaiyi Huang, Kaiyue Sun, Enze Xie, Zhenguo Li, and Xihui Liu. T2i-compbench: A comprehensive benchmark for open-world compositional text-to-image generation. *Advances in Neural Information Processing Systems*, 36:78723–78747, 2023.

Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *International conference on machine learning*, pp. 12888–12900. PMLR, 2022.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. Dora: Weight-decomposed low-rank adaptation. *arXiv preprint arXiv:2402.09353*, 2024.

Ilya Loshchilov, Frank Hutter, et al. Fixing weight decay regularization in adam. *arXiv preprint arXiv:1711.05101*, 5, 2017.

Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. Sdxl: Improving latent diffusion models for high-resolution image synthesis. *arXiv preprint arXiv:2307.01952*, 2023.

- Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PMLR, 2021.
- Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10684–10695, 2022.
- Yi-Lin Sung, Varun Nair, and Colin A Raffel. Training neural networks with fixed sparse masks. *Advances in Neural Information Processing Systems*, 34:24193–24205, 2021.
- Thanh Van Le, Hao Phung, Thuan Hoang Nguyen, Quan Dao, Ngoc N Tran, and Anh Tran. Anti-dreambooth: Protecting users from personalized text-to-image synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 2116–2127, 2023.