

PatchDPO: Patch-level DPO for Finetuning-free Personalized Image Generation

–Supplementary Material–

S1 Proof

As introduced in the main paper, PatchDPO estimates the patch quality of the images, $p(\mathbf{x}_{\text{ref}}) \in \mathbb{R}^{H \times W}$ and $p(\mathbf{x}_{\text{gen}}) \in \mathbb{R}^{H \times W}$, and accordingly leverages a weighted training method for model optimization using the estimated patch quality. Specifically, PatchDPO trains the original personalized generation model with an image reconstruction task, then assigns higher training weights to the patches with higher quality, while assigning lower training weights to the patches with lower quality. Finally, the loss function of PatchDPO (with the task of reconstructing \mathbf{x}_{ref} with \mathbf{x}_{gen}) is calculated as below:

$$\begin{aligned} \mathcal{L}_{\text{PatchDPO}} = & \underbrace{\| [\epsilon_{\text{gen}} - \epsilon_{\theta}(\mathbf{x}_{\text{gen}}(t), \mathbf{c}_{\text{text}}, \mathbf{x}_{\text{ref}}, t)] \odot \tilde{p}(\mathbf{x}_{\text{gen}}) \|_2^2}_{\text{Reconstruct } \mathbf{x}_{\text{gen}} \text{ with } \mathbf{x}_{\text{ref}}} \\ & + \underbrace{\| [\epsilon_{\text{ref}} - \epsilon_{\theta}(\mathbf{x}_{\text{ref}}(t), \mathbf{c}_{\text{text}}, \mathbf{x}_{\text{ref}}, t)] \odot (1 - \tilde{p}(\mathbf{x}_{\text{ref}})) \|_2^2}_{\text{Reconstruct } \mathbf{x}_{\text{ref}} \text{ with } \mathbf{x}_{\text{ref}}}. \end{aligned}$$

In this section, we prove that assigning higher training weights to the patches could facilitate the generation of these patches. Note that we complete this proof in a simplified case (with a linear layer) without loss of generalization, since the linear layers are widely present in the UNet of the diffusion model.

Definition: Let $\mathbf{x} \in \mathbb{R}^M$ denote the original image (the image is flattened for convenience), $\tilde{\mathbf{x}} \in \mathbb{R}^M$ denote the noisy image, ϵ_{θ} denote the model for noise prediction with $\mathbf{W} \in \mathbb{R}^{M \times M}$ as weight, $p \in [0, 1]^M$ denote the patch quality (corresponding to each element in \mathbf{x}), then $\epsilon = \tilde{\mathbf{x}} - \mathbf{x}$ is the noise to be predicted ($\epsilon \in \mathbb{R}^M$), and the loss for image reconstruction is $\mathcal{L} = \|p(\epsilon_{\theta}(\tilde{\mathbf{x}}) - \epsilon)\|_2^2 = \|p(\mathbf{W}\tilde{\mathbf{x}} - \epsilon)\|_2^2 = \sum_{i=1}^M p_i^2 (\mathbf{W}_i \tilde{\mathbf{x}} - \epsilon_i)^2$ (note that $\mathbf{c}_{\text{text}}, \mathbf{x}_{\text{ref}}, t$ are omitted here for simplicity).

Theorem: For each element i , the absolute value of distance between the updated difference $(\hat{\epsilon}_{\theta}(\tilde{\mathbf{x}})_i - \epsilon_i)^2$ and the original difference $(\epsilon_{\theta}(\tilde{\mathbf{x}})_i - \epsilon_i)^2$ is positively correlated with p_i ($\hat{\epsilon}_{\theta}$ is the updated model with $\hat{\mathbf{W}}$ as weight, η is the learning rate).

Proof. According to $\mathcal{L} = \|p(\epsilon_{\theta}(\tilde{\mathbf{x}}) - \epsilon)\|_2^2 = \|p(\mathbf{W}\tilde{\mathbf{x}} - \epsilon)\|_2^2 = \sum_{i=1}^M p_i^2 (\mathbf{W}_i \tilde{\mathbf{x}} - \epsilon_i)^2$, then:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}_{ij}} = 2p_i^2 (\mathbf{W}_i \tilde{\mathbf{x}} - \epsilon_i) \tilde{\mathbf{x}}_j. \quad (1)$$

Therefore, at each step of gradient descent, each element $\hat{\mathbf{W}}_{ij}$ of $\hat{\mathbf{W}}$ is updated as below:

$$\begin{aligned} \hat{\mathbf{W}}_{ij} &= \mathbf{W}_{ij} - \eta \frac{\partial \mathcal{L}}{\partial \mathbf{W}_{ij}} \\ &= \mathbf{W}_{ij} - 2\eta p_i^2 (\mathbf{W}_i \tilde{\mathbf{x}} - \epsilon_i) \tilde{\mathbf{x}}_j. \end{aligned} \quad (2)$$

Next, for each element i , $\Delta_i = (\hat{\epsilon}_{\theta}(\tilde{\mathbf{x}})_i - \epsilon_i)^2 - (\epsilon_{\theta}(\tilde{\mathbf{x}})_i - \epsilon_i)^2$ is calculated as below:

$$\begin{aligned} \Delta_i &= (\hat{\epsilon}_{\theta}(\tilde{\mathbf{x}})_i - \epsilon_i)^2 - (\epsilon_{\theta}(\tilde{\mathbf{x}})_i - \epsilon_i)^2 \\ &= (\hat{\mathbf{W}}_i \tilde{\mathbf{x}} - \epsilon_i)^2 - (\mathbf{W}_i \tilde{\mathbf{x}} - \epsilon_i)^2 \\ &= \left(\sum_{j=1}^M \hat{\mathbf{W}}_{ij} \tilde{\mathbf{x}}_j - \epsilon_i \right)^2 - \left(\sum_{j=1}^M \mathbf{W}_{ij} \tilde{\mathbf{x}}_j - \epsilon_i \right)^2 \\ &= \left[\sum_{j=1}^M (\mathbf{W}_{ij} - 2\eta p_i^2 (\mathbf{W}_i \tilde{\mathbf{x}} - \epsilon_i) \tilde{\mathbf{x}}_j) \tilde{\mathbf{x}}_j - \epsilon_i \right]^2 - \left(\sum_{j=1}^M \mathbf{W}_{ij} \tilde{\mathbf{x}}_j - \epsilon_i \right)^2 \end{aligned} \quad (3)$$

$$\begin{aligned}
&= \left[\sum_{j=1}^M \mathbf{W}_{ij} \tilde{\mathbf{x}}_j - 2\eta p_i^2 (\mathbf{W}_i \tilde{\mathbf{x}} - \boldsymbol{\epsilon}_i) \sum_{j=1}^M \tilde{\mathbf{x}}_j^2 - \boldsymbol{\epsilon}_i \right]^2 - (\mathbf{W}_i \tilde{\mathbf{x}} - \boldsymbol{\epsilon}_i)^2 \\
&= \left[\mathbf{W}_i \tilde{\mathbf{x}} - \boldsymbol{\epsilon}_i - 2\eta p_i^2 (\mathbf{W}_i \tilde{\mathbf{x}} - \boldsymbol{\epsilon}_i) \sum_{j=1}^M \tilde{\mathbf{x}}_j^2 \right]^2 - (\mathbf{W}_i \tilde{\mathbf{x}} - \boldsymbol{\epsilon}_i)^2 \\
&= \left[(1 - 2\eta p_i^2 \sum_{j=1}^M \tilde{\mathbf{x}}_j^2) (\mathbf{W}_i \tilde{\mathbf{x}} - \boldsymbol{\epsilon}_i) \right]^2 - (\mathbf{W}_i \tilde{\mathbf{x}} - \boldsymbol{\epsilon}_i)^2 \\
&= (1 - 2\eta p_i^2 \sum_{j=1}^M \tilde{\mathbf{x}}_j^2)^2 (\mathbf{W}_i \tilde{\mathbf{x}} - \boldsymbol{\epsilon}_i)^2 - (\mathbf{W}_i \tilde{\mathbf{x}} - \boldsymbol{\epsilon}_i)^2.
\end{aligned} \tag{4}$$

Note that the value of learning rate η is small enough to ensure the decrease of loss \mathcal{L} (i.e., $0 \leq 1 - 2\eta p_i^2 \sum_{j=1}^M \tilde{\mathbf{x}}_j^2 \leq 1$, and $\Delta_i \leq 0$). Therefore, increasing p_i will decrease $(1 - 2\eta p_i^2 \sum_{j=1}^M \tilde{\mathbf{x}}_j^2)^2$ and Δ_i , thus increasing $\|\Delta_i\|$. Finally, $\|\Delta_i\|$ is positively correlated with p_i . The proof is ended here. \square

This theorem indicates that in PatchDPO, the generation of high-quality patches is rewarded using the high training weights, while the generation of low-quality patches is punished using the low training weights.

S2 Experiments

S2.1 Implementation Details

S2.1.1 Multi-Object Generation

Dataset. The multi-object dataset for the PatchDPO training also consists of 50,000 images, which is constructed in a similar manner as the single-object dataset. Detailedly, for the multi-object dataset, we utilize ChatGPT to generate the text prompts in the format of “An $\{object\ 1\}$ and an $\{object\ 2\}$ in the $\{background\}$ ”. Besides, when feeding the text prompts into Stable Diffusion [10], we also instruct it to generate the non-overlapped objects to avoid confusion between objects.

Model. Our main experiments are conducted on the pre-trained IP-Adapter-Plus [15], which adopts the decoupled cross-attention mechanism to merge the text features and reference image features into the middle layers of diffusion model ϵ_θ . Specifically, in the decoupled cross-attention mechanism, the latent image features $\mathbf{Z} \in \mathbb{R}^{(H_\epsilon \cdot W_\epsilon) \times D_\epsilon}$ in a middle layer are fed into a cross-attention module to interact with the text features $\mathbf{c}_{\text{text}} \in \mathbb{R}^{S_{\text{text}} \times D_{\text{text}}}$ (extracted with a text encoder):

$$\mathbf{Z}_{\text{text}} = \text{Attn}(\mathbf{Q}, \mathbf{K}_{\text{text}}, \mathbf{V}_{\text{text}}) = \text{Softmax}\left(\frac{\mathbf{Q}\mathbf{K}_{\text{text}}^\top}{\sqrt{d}}\right)\mathbf{V}_{\text{text}}. \tag{5}$$

Here, $\mathbf{Q} = \mathbf{Z}\mathbf{W}^\mathbf{Q}$, $\mathbf{K}_{\text{text}} = \mathbf{c}_{\text{text}}\mathbf{W}_{\text{text}}^\mathbf{K}$, $\mathbf{V}_{\text{text}} = \mathbf{c}_{\text{text}}\mathbf{W}_{\text{text}}^\mathbf{V}$ are the query, key, and value matrices of the attention operation, respectively, and $\mathbf{W}^\mathbf{Q} \in \mathbb{R}^{D_\epsilon \times D_\epsilon}$, $\mathbf{W}_{\text{text}}^\mathbf{K} \in \mathbb{R}^{D_{\text{text}} \times D_\epsilon}$, $\mathbf{W}_{\text{text}}^\mathbf{V} \in \mathbb{R}^{D_{\text{text}} \times D_\epsilon}$ are the learnable weight matrices for feature projection. Besides, \mathbf{Z} is also fed into another cross-attention module to interact with the reference image features $\mathbf{c}_{\text{img}}^i \in \mathbb{R}^{S_{\text{img}} \times D_{\text{img}}}$ (extracted with an image encoder) for the i -th reference image:

$$\mathbf{Z}_{\text{img}}^i = \text{Attn}(\mathbf{Q}, \mathbf{K}_{\text{img}}^i, \mathbf{V}_{\text{img}}^i) = \text{Softmax}\left(\frac{\mathbf{Q}\mathbf{K}_{\text{img}}^{i\top}}{\sqrt{d}}\right)\mathbf{V}_{\text{img}}^i. \tag{6}$$

Likewise, $\mathbf{K}_{\text{img}}^i = \mathbf{c}_{\text{img}}^i\mathbf{W}_{\text{img}}^\mathbf{K}$, $\mathbf{V}_{\text{img}}^i = \mathbf{c}_{\text{img}}^i\mathbf{W}_{\text{img}}^\mathbf{V}$, and $\mathbf{W}_{\text{img}}^\mathbf{K} \in \mathbb{R}^{D_{\text{img}} \times D_\epsilon}$, $\mathbf{W}_{\text{img}}^\mathbf{V} \in \mathbb{R}^{D_{\text{img}} \times D_\epsilon}$ are the learnable weight matrices for projecting the reference image features. Next, for M reference images provided in *multi-object generation*, the final output \mathbf{Z}_{new} of the decoupled cross-attention mechanism is the addition of \mathbf{Z}_{text} and $\{\mathbf{Z}_{\text{img}}^i\}_{i=1}^M$:

$$\mathbf{Z}_{\text{new}} = \mathbf{Z}_{\text{text}} + \sum_{i=1}^M \mathbf{Z}_{\text{img}}^i. \tag{7}$$

Method	DINO	CLIP-I	CLIP-T	Avg.
DreamBooth • [11]	0.3849	0.6636	0.7383	0.5956
Custom Diffusion (Opt) • [5]	0.3684	0.6595	0.7599	0.5959
Custom Diffusion (Joint) • [5]	0.3799	0.6704	0.7534	0.6012
Mix-of-Show § [3]	0.3940	0.6700	0.7280	0.5973
MC ² § [4]	0.4060	0.6860	0.7670	0.6197
FastComposer ★ [14]	0.3574	0.6552	0.7456	0.5861
λ-ECLIPSE ★ [8]	0.3902	0.6902	0.7275	0.6026
ELITE ★ [13]	0.3347	0.6460	0.6814	0.5540
IP-Adapter-Plus ★ [15]	0.3992	0.6904	0.7655	0.6184
SSR-Encoder ★ [17]	0.3970	0.6895	0.7363	0.6076
PatchDPO	0.4168	0.6945	0.7726	0.6280

Table 1. Performance comparison for multi-object personalized generation on *Concept101*. The upper methods are finetuning-based methods, the bottom methods are finetuning-free methods, and bold font denotes the best result. Each CLIP-T score is multiplied by 2.5 following Custom Diffusion.

Method	DINO	CLIP-I	CLIP-T	Avg.
DreamBooth † [11]	0.430	0.695	0.308	0.478
Custom Diffusion † [5]	0.464	0.698	0.300	0.487
Subject-Diffusion † [5]	0.506	0.696	0.310	0.504
IP-Adapter-Plus ★ [15]	0.468	0.683	0.315	0.489
PatchDPO	0.506	0.705	0.322	0.511

Table 2. Performance comparison for multi-object personalized generation on *MultiDreamBench*. The results of methods marked with † are from the paper of Subject-Diffusion [7], and the results of methods marked with ★ are re-implemented faithfully following their released code and weights. Bold font denotes the best result.

Method	DINO	CLIP-I	CLIP-T	Avg.
IP-Adapter [15]	0.608	0.809	0.274	0.564
IP-Adapter (PatchDPO)	0.625	0.817	0.281	0.574
ELITE [13]	0.652	0.762	0.255	0.556
ELITE (PatchDPO)	0.678	0.776	0.264	0.573
IP-Adapter-Plus [15]	0.692	0.826	0.281	0.600
IP-Adapter-Plus (PatchDPO)	0.727	0.838	0.292	0.619

Table 3. Ablation experiments of PatchDPO for different pre-trained personalized generation models on *DreamBench*.

In this manner, the information of multiple reference images is incorporated into the diffusion model to generate the corresponding images.

S2.1.2 Diffusion-DPO

Direct preference optimization [9] simplifies RLHF by training the model directly from human preferences, and the DPO loss \mathcal{L}_{DPO} is calculated as below:

$$\mathcal{L}_{\text{DPO}} = -\mathbb{E}_{\mathbf{c}, \mathbf{x}^w, \mathbf{x}^l} \left[\log \sigma \left(\beta \log \frac{p_{\theta}(\mathbf{x}^w | \mathbf{c})}{p_{\text{ref}}(\mathbf{x}^w | \mathbf{c})} - \beta \log \frac{p_{\theta}(\mathbf{x}^l | \mathbf{c})}{p_{\text{ref}}(\mathbf{x}^l | \mathbf{c})} \right) \right], \quad (8)$$

where $\sigma(\cdot)$ denotes the sigmoid function, p_{ref} denotes the original model, \mathbf{x}^w and \mathbf{x}^l denote the “winning” and “losing” samples generated from the condition \mathbf{c} . DPO optimizes the model by aligning its output closer to \mathbf{x}^w while distancing it from \mathbf{x}^l . Diffusion-DPO [12] adapts this original DPO loss to the diffusion model:

Method	DINO	CLIP-I	CLIP-T
Re-Imagen [1]	0.600	0.740	0.270
λ -ECLIPSE [8]	0.613	0.783	0.307
ELITE [13]	0.652	0.762	0.255
IP-Adapter [15]	0.608	0.809	0.274
IP-Adapter-Plus [15]	0.692	0.826	0.281
SSR-Encoder [17]	0.612	0.821	0.308
BLIP-Diffusion [6]	0.594	0.779	0.300
Subject-Diffusion [7]	0.711	0.787	0.293
JeDi [16]	0.679	0.814	0.293
PatchDPO	0.727 \pm 0.002	0.838 \pm 0.002	0.292 \pm 0.002

Table 4. Performance comparison (with standard deviation) for single-object personalized generation on *DreamBench*.

Method	DINO	CLIP-I	CLIP-T
FastComposer \star [14]	0.3574 \pm 0.0025	0.6552 \pm 0.0015	0.7456 \pm 0.0019
λ -ECLIPSE \star [8]	0.3902 \pm 0.0019	0.6902 \pm 0.0007	0.7275 \pm 0.0017
ELITE \star [13]	0.3347 \pm 0.0022	0.6460 \pm 0.0011	0.6814 \pm 0.0026
IP-Adapter-Plus \star [15]	0.3992 \pm 0.0045	0.6904 \pm 0.0019	0.7655 \pm 0.0033
SSR-Encoder \star [17]	0.3970 \pm 0.0027	0.6895 \pm 0.0016	0.7363 \pm 0.0020
PatchDPO	0.4168 \pm 0.0024	0.6945 \pm 0.0026	0.7726 \pm 0.0027

Table 5. Performance comparison (with standard deviation) for multi-object personalized generation on *Concept101*. Each CLIP-T score is multiplied by 2.5 following Custom Diffusion.

$$\mathcal{L}_{\text{Diffusion-DPO}} = -\mathbb{E}_{(\mathbf{x}_0^w, \mathbf{x}_0^l) \sim \mathcal{D}, t \sim \mathcal{U}(0, T), \mathbf{x}_t^w \sim q(\mathbf{x}_t^w | \mathbf{x}_0^w), \mathbf{x}_t^l \sim q(\mathbf{x}_t^l | \mathbf{x}_0^l)} \log \sigma(-\beta T \omega(\lambda_t) (\|\boldsymbol{\epsilon}^w - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t^w, t)\|_2^2 - \|\boldsymbol{\epsilon}^w - \boldsymbol{\epsilon}_{\text{ref}}(\mathbf{x}_t^w, t)\|_2^2 - (\|\boldsymbol{\epsilon}^l - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t^l, t)\|_2^2 - \|\boldsymbol{\epsilon}^l - \boldsymbol{\epsilon}_{\text{ref}}(\mathbf{x}_t^l, t)\|_2^2))). \quad (9)$$

We utilize this Diffusion-DPO loss to train the model for comparison with PatchDPO. Specifically, the image preference (“winning” sample v.s “losing” sample) required for DPO training is estimated by comparing the image similarity generated by the ViT-Base model [2]. As shown in **Table 4** of the main paper, Combination (3) (Diffusion-DPO) fails to improve the performance of the original model, because Diffusion-DPO would wrongly reward the low-quality patches in the “winning” sample, while wrongly punishing the high-quality patches in the “losing” sample. Instead, Combination (4) (PatchDPO) correctly rewards the high-quality patches and punishes the low-quality patches, thus achieving a huge performance improvement.

S2.2 Multi-Object Personalized Generation

Table 1 demonstrates the quantitative results of different methods on *Concept101*. Note that the results of methods marked with \bullet are from the GitHub page of Custom Diffusion [5], the results of methods marked with \S are from the paper of MC² [4], and the results of methods marked with \star are re-implemented faithfully following their released code and weights (their original evaluation datasets have not been made public).

MultiDreamBench is a benchmark for the evaluation of multi-object personalized image generation, by combining the reference images from the original *DreamBench* [11] for generation. *MultiDreamBench* is proposed by Subject-Diffusion [7], and we evaluate our model on it for comparison with Subject-Diffusion. As shown in **Table 2**, PatchDPO can also improve the performance of original IP-Adapter-Plus, and surpass existing personalized generation methods on *MultiDreamBench*.

S2.3 More Ablation Experiments

Different pre-trained models. Besides IP-Adapter-Plus, this work also employs PatchDPO on other pre-trained personalized generation models (*i.e.*, ELITE [13], original IP-Adapter [15]). As shown in **Table 3**, PatchDPO can also improve the performance of ELITE and the original IP-Adapter on *DreamBench*, exhibiting its strong capability for generalization.

S2.4 Computational Cost

PatchDPO continues to train the pre-trained personalized generation model, eliminating the need to retrain the model from scratch. Besides, PatchDPO utilizes a small-sized but high-quality dataset (consisting of 50,000 images) for training. Consequently, PatchDPO is computationally efficient compared to the original pre-training stage. Specifically, the PatchDPO training for single-object personalized generation on IP-Adapter-Plus takes only 4.30 Hours, and the PatchDPO training for multi-object personalized generation on IP-Adapter-Plus takes only 6.23 Hours.

S2.5 Standard Deviation of Performance

In this section, we provide the standard deviation of performance on both *DreamBench* and *Concept101* of the main paper. As shown in [Table 4](#) and [Table 5](#), our model can achieve relatively stable performance, compared to other methods.

S3 More Visualizations

Natural images & generated images. [Figure 1](#) demonstrates the comparison between natural images (from $\mathcal{D}_{\text{natural}}$) and our generated images (from $\mathcal{D}_{\text{ours}}$), indicating that natural images contain objects of complex details and have confusing objects and backgrounds, which seriously hinder model training. As shown in [Table 4](#) of the main paper, Combination (1) (using $\mathcal{D}_{\text{natural}}$) seriously degrades the image-alignment (DINO, CLIP-I) of model, and Combination (2) (using $\mathcal{D}_{\text{ours}}$) benefits the model performance.

Personalized image generation with human image. [Figure 2](#) presents the visualization results of single-object personalized image generation with the human image as reference (5 famous persons and 2 persons from *Concept101*), implying that our method performs well in preserving the identity details of human faces.

Single-object personalized image generation. [Figure 3](#) demonstrates more qualitative results of different methods on *DreamBench*. Compared to existing methods, PatchDPO exhibits superior performance in preserving the local details of the reference image, resulting in the generation of higher quality.

Multi-object personalized image generation. [Figure 4](#) demonstrates more qualitative results of different methods on *Concept101*, validating the outstanding performance of PatchDPO on multi-object personalized image generation.



Figure 1. Comparison between natural images (from $\mathcal{D}_{\text{natural}}$) and our generated images (from $\mathcal{D}_{\text{ours}}$).

References

- [1] Wenhu Chen, Hexiang Hu, Chitwan Saharia, and William W. Cohen. Re-imagen: Retrieval-augmented text-to-image generator. In *ICLR 2023*. OpenReview.net, 2023. 4
- [2] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR 2021*. OpenReview.net, 2021. 4
- [3] Yuchao Gu, Xintao Wang, Jay Zhangjie Wu, Yujun Shi, Yunpeng Chen, Zihan Fan, Wuyou Xiao, Rui Zhao, Shuning Chang, Weijia Wu, Yixiao Ge, Ying Shan, and Mike Zheng Shou. Mix-of-show: Decentralized low-rank adaptation for multi-concept customization of diffusion models. In *NeurIPS 2023*, 2023. 3
- [4] Jiaxiu Jiang, Yabo Zhang, Kailai Feng, Xiaohe Wu, and Wangmeng Zuo. Mc²: Multi-concept guidance for customized multi-concept generation. *arXiv preprint arXiv:2404.05268*, 2024. 3, 4
- [5] Nupur Kumari, Bingliang Zhang, Richard Zhang, Eli Shechtman, and Jun-Yan Zhu. Multi-concept customization of text-to-image diffusion. In *CVPR 2023*, pages 1931–1941. IEEE, 2023. 3, 4
- [6] Dongxu Li, Junnan Li, and Steven C. H. Hoi. Blip-diffusion: Pre-trained subject representation for controllable text-to-image generation and editing. In *NeurIPS 2023*, 2023. 4
- [7] Jian Ma, Junhao Liang, Chen Chen, and Haonan Lu. Subject-diffusion: Open domain personalized text-to-image generation without test-time fine-tuning. In *SIGGRAPH 2024*, page 25. ACM, 2024. 3, 4
- [8] Maitreya Patel, Sangmin Jung, Chitta Baral, and Yezhou Yang. *lambda*-eclipse: Multi-concept personalized text-to-image diffusion models by leveraging clip latent space. *arXiv preprint arXiv:2402.05195*, 2024. 3, 4
- [9] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *NeurIPS 2023*, 2023. 3
- [10] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR 2022*, pages 10674–10685. IEEE, 2022. 2
- [11] Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. In *CVPR 2023*, pages 22500–22510. IEEE, 2023. 3, 4
- [12] Bram Wallace, Meihua Dang, Rafael Rafailov, Linqi Zhou, Aaron Lou, Senthil Purushwalkam, Stefano Ermon, Caiming Xiong, Shafiq Joty, and Nikhil Naik. Diffusion model alignment using direct preference optimization. In *CVPR 2024*, pages 8228–8238. IEEE, 2024. 3
- [13] Yuxiang Wei, Yabo Zhang, Zhilong Ji, Jinfeng Bai, Lei Zhang, and Wangmeng Zuo. ELITE: encoding visual concepts into textual embeddings for customized text-to-image generation. In *ICCV 2023*, pages 15897–15907. IEEE, 2023. 3, 4
- [14] Guangxuan Xiao, Tianwei Yin, William T Freeman, Frédo Durand, and Song Han. Fastcomposer: Tuning-free multi-subject image generation with localized attention. *arXiv preprint arXiv:2305.10431*, 2023. 3, 4
- [15] Hu Ye, Jun Zhang, Sibio Liu, Xiao Han, and Wei Yang. Ip-adapter: Text compatible image prompt adapter for text-to-image diffusion models. *arXiv preprint arXiv:2308.06721*, 2023. 2, 3, 4
- [16] Yu Zeng, Vishal M. Patel, Haochen Wang, Xun Huang, Ting-Chun Wang, Ming-Yu Liu, and Yogesh Balaji. Jedi: Joint-image diffusion models for finetuning-free personalized text-to-image generation. In *CVPR 2024*, pages 6786–6795. IEEE, 2024. 4
- [17] Yuxuan Zhang, Yiren Song, Jiaming Liu, Rui Wang, Jinpeng Yu, Hao Tang, Huaxia Li, Xu Tang, Yao Hu, Han Pan, et al. Ssr-encoder: Encoding selective subject representation for subject-driven generation. In *CVPR 2024*, pages 8069–8078. IEEE, 2024. 3, 4

Reference Image	Prompt	Custom Diffusion	IP-Adapter	ELITE	SSR-Encoder	PatchDPO
	A man on the beach.					
	A man on the beach.					
	A man on the beach.					
	A man on the beach.					
	A man on the beach.					
	A man on the beach.					
	A man on the beach.					

Figure 2. Qualitative comparisons of different methods on personalized image generation with the human image as reference.

























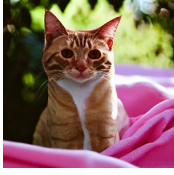




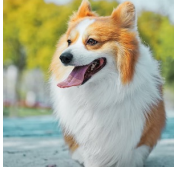
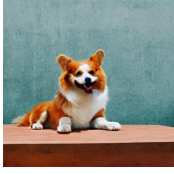
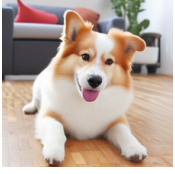
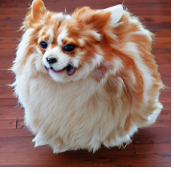
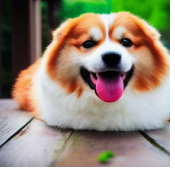
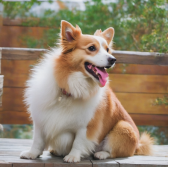
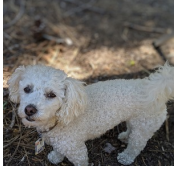
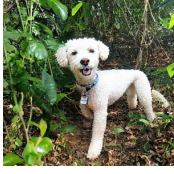
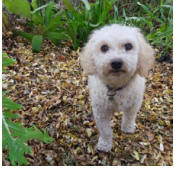
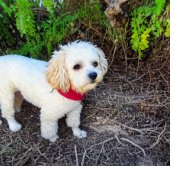
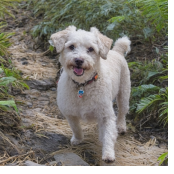
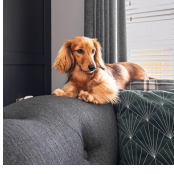
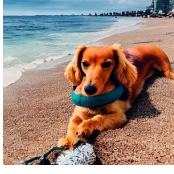
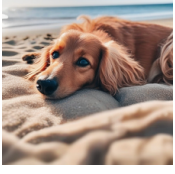

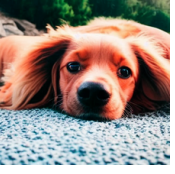
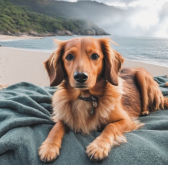
Reference Image	Prompt	Custom Diffusion	IP-Adapter	ELITE	SSR-Encoder	PatchDPO
	A clock in the snow.					
	A boot with a city in the background.					
	A toy on top of a purple rug in a forest.					
	A sneaker with a tree and autumn leaves in the background.					
	A cat on top of pink fabric.					
	A dog on top of a wooden floor.					
	A dog in the jungle.					
	A dog on the beach.					

Figure 3. Qualitative comparisons of different methods on single-object personalized image generation.

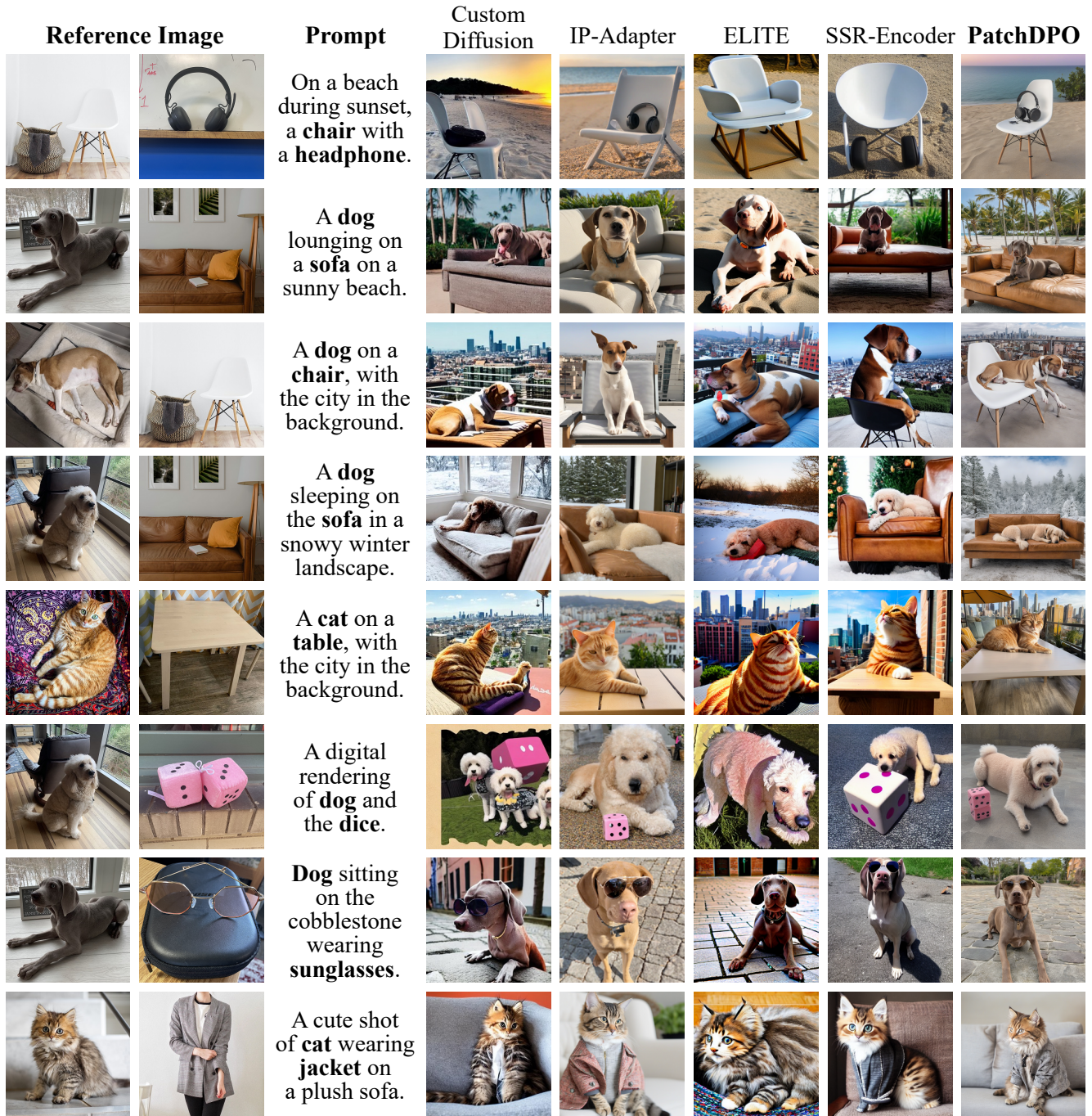


Figure 4. Qualitative comparisons of different methods on multi-object personalized image generation.