Efficient Personalization of Quantized Diffusion Model without Backpropagation

Supplementary Materials

Hoigi Seo[*1] Wongi Jeong[*1] Kyungryeol Lee[1] Se Young Chun[1,2†]
[1]Dept. of Electrical and Computer Engineering, [2]INMC & IPAI
Seoul National University, Republic of Korea
{seohoiki3215, wg7139, kr.lee, sychun}@snu.ac.kr

S1. Additional Experimental Details

S1.1. Dataset

All quantitative experiments were conducted using the DreamBooth [14] (DB) dataset, which includes 30 distinct subjects which are denoted in Tab. S1, each paired with 25 unique prompts. Among these subjects, nine are living entities—specifically dogs and cats—while the remaining 21 represent non-living objects. Details of the subjects and their respective prompts are provided in the Tab. S2 (living) and Tab. S3 (non-living). Notably, the Textual Inversion [4]-based method did not incorporate class tokens in the prompts used for image generation, ensuring a fair comparison by excluding scenarios where class tokens are utilized during the unique token learning process. For Textual Inversion [4] (TI) and Gradient-Free Textual Inversion [3] (GF-TI), we employed the ImageNet-based template proposed in the original TI paper for training. ZOODiP, followed the DCO [6], utilizing Comprehensive Captioning (CC) for the text prompt.

S1.2. Metrics

To compute the CLIP-I score, we use the `openai/clip-vit-base-patch32` model [12] from Huggingface to extract image features for both the reference and generated images. We then calculate the cosine similarity for all possible pairs and average these values. For the CLIP-T score, we leverage the same model's text encoder to extract text features from the input prompt. We also calculate image features for the generated images and measure the pairwise cosine similarity between the text and image features, averaging the results to obtain the final score. To compute the DINO score, we utilize the `facebook/dinov2-base` model from Huggingface to extract DINOv2 [9] embeddings for both the reference and generated images. The score is determined by averaging the pairwise cosine similarities between these embeddings.

S1.3. Quantization

We utilized `optimum-quanto`, the PyTorch [10] quantization backend provided by Huggingface, to enable accelerated matrix multiplications on CUDA devices. For the 8-bit weight quantization described in the main paper, we employed `optimum-quanto` to quantize the weights of all Linear and Conv2D layers in the VAE, U-Net, and text encoder modules to 8-bit integer. Other layers' parameters including text embedding, layer normalization, batch normalization were not quantized. With this process, 96.4% of whole Stable Diffusion [13] pipeline (U-Net, VAE, and text encoder) parameters are quantized to INT8, while other 3.6% parameters are remained FP16.

S1.4. Memory usage

We primarily monitored memory usage using the `nvidia-smi` command to observe real-time memory consumption during training and employed the PyTorch profiler for detailed memory breakdowns. Notably, the memory utilized by the CUDA context can vary based on the CUDA and PyTorch version, so the reported memory usage may differ depending on the experimental environment. Our measurements were conducted under the following settings: CUDA Toolkit 11.8, Torch 2.4.1, Torchvision 0.19.1, and Python 3.8.10. Since PyTorch loads all CUDA libraries by default, including unused ones, the actual memory usage could potentially be reduced by unloading unnecessary libraries. However, as our focus was not on such optimizations, we ensured a fair comparison by evaluating all training methods under the same experimental environment. We also observed that Variational Auto-Encoder (VAE) encoding introduces significant memory overhead. Given that all methods do not require backpropagation through the VAE, we standardized the process by blocking gradients in the VAE

[*] Authors contributed equally. † Corresponding author.

Subjects in DreamBooth [14] dataset
backpack, backpack_dog, bear_plushie, berry_bowl, can, candle, cat, cat2, clock, colorful_sneaker, dog, dog2, dog3, dog5, dog6, dog7, dog8, duck_toy, fancy_boot, grey_sloth_plushie, monster_toy, pink_sunglasses, poop_emoji, rc_car, red_cartoon, robot_toy, shiny_sneaker, teapot, vase, wolf_plushie

Table S1. Full subjects name of DreamBooth dataset. The dataset names in the main paper are based on the corresponding subject datasets.

	Full prompt used in evaluation (living)
1	`'a {0} {1} in the jungle'.format(unique_token, class_token)`
2	`'a {0} {1} in the snow'.format(unique_token, class_token)`
3	`'a {0} {1} on the beach'.format(unique_token, class_token)`
4	`'a {0} {1} on a cobblestone street'.format(unique_token, class_token)`
5	`'a {0} {1} on top of pink fabric'.format(unique_token, class_token)`
6	`'a {0} {1} on top of a wooden floor'.format(unique_token, class_token)`
7	`'a {0} {1} with a city in the background'.format(unique_token, class_token)`
8	`'a {0} {1} with a mountain in the background'.format(unique_token, class_token)`
9	`'a {0} {1} with a blue house in the background'.format(unique_token, class_token)`
10	`'a {0} {1} on top of a purple rug in a forest'.format(unique_token, class_token)`
11	`'a {0} {1} wearing a red hat'.format(unique_token, class_token)`
12	`'a {0} {1} wearing a santa hat'.format(unique_token, class_token)`
13	`'a {0} {1} wearing a rainbow scarf'.format(unique_token, class_token)`
14	`'a {0} {1} wearing a black top hat and a monocle'.format(unique_token, class_token)`
15	`'a {0} {1} in a chef outfit'.format(unique_token, class_token)`
16	`'a {0} {1} in a firefighter outfit'.format(unique_token, class_token)`
17	`'a {0} {1} in a police outfit'.format(unique_token, class_token)`
18	`'a {0} {1} wearing pink glasses'.format(unique_token, class_token)`
19	`'a {0} {1} wearing a yellow shirt'.format(unique_token, class_token)`
20	`'a {0} {1} in a purple wizard outfit'.format(unique_token, class_token)`
21	`'a red {0} {1}'.format(unique_token, class_token)`
22	`'a purple {0} {1}'.format(unique_token, class_token)`
23	`'a shiny {0} {1}'.format(unique_token, class_token)`
24	`'a wet {0} {1}'.format(unique_token, class_token)`
25	`'a cube shaped {0} {1}'.format(unique_token, class_token)`

Table S2. Full prompts used in evaluation of living category objects. `unique_token` represents the special token corresponds to object which aims to personalize, and `class_token` denotes the class that `unique_token` is in.

	Full prompt used in evaluation (non-living)
1	`'a {0} {1} in the jungle'.format(unique_token, class_token)`
2	`'a {0} {1} in the snow'.format(unique_token, class_token)`
3	`'a {0} {1} on the beach'.format(unique_token, class_token)`
4	`'a {0} {1} on a cobblestone street'.format(unique_token, class_token)`
5	`'a {0} {1} on top of pink fabric'.format(unique_token, class_token)`
6	`'a {0} {1} on top of a wooden floor'.format(unique_token, class_token)`
7	`'a {0} {1} with a city in the background'.format(unique_token, class_token)`
8	`'a {0} {1} with a mountain in the background'.format(unique_token, class_token)`
9	`'a {0} {1} with a blue house in the background'.format(unique_token, class_token)`
10	`'a {0} {1} on top of a purple rug in a forest'.format(unique_token, class_token)`
11	`'a {0} {1} with a wheat field in the background'.format(unique_token, class_token)`
12	`'a {0} {1} with a tree and autumn leaves in the background'.format(unique_token, class_token)`
13	`'a {0} {1} with the Eiffel Tower in the background'.format(unique_token, class_token)`
14	`'a {0} {1} floating on top of water'.format(unique_token, class_token)`
15	`'a {0} {1} floating in an ocean of milk'.format(unique_token, class_token)`
16	`'a {0} {1} on top of green grass with sunflowers around it'.format(unique_token, class_token)`
17	`'a {0} {1} on top of a mirror'.format(unique_token, class_token)`
18	`'a {0} {1} on top of the sidewalk in a crowded street'.format(unique_token, class_token)`
19	`'a {0} {1} on top of a dirt road'.format(unique_token, class_token)`
20	`'a {0} {1} on top of a white rug'.format(unique_token, class_token)`
21	`'a red {0} {1}'.format(unique_token, class_token)`
22	`'a purple {0} {1}'.format(unique_token, class_token)`
23	`'a shiny {0} {1}'.format(unique_token, class_token)`
24	`'a wet {0} {1}'.format(unique_token, class_token)`
25	`'a cube shaped {0} {1}'.format(unique_token, class_token)`

Table S3. Full prompts used in evaluation of non-living category objects. `unique_token` represents the special token corresponds to object which aims to personalize, and `class_token` denotes the class that `unique_token` is in.

Method	CLIP-T↑	CLIP-I↑	DINO↑
RGE	0.266	0.759	0.569
SPSA	0.277	0.732	0.506
One-point	0.296	0.703	0.393

Table S4. Personalization performance across different gradient estimation methods with $n = 2$. The results show that RGE outperforms other two different gradient estimations methods. Notably, RGE is more efficient in terms of computational cost, requiring fewer forward passes than SPSA when $n > 1$. Due to this efficiency and performance advantages, we adopted RGE as the gradient estimation method in ZOODiP.

encoding step across all methods. This adjustment ensured that each personalization method used the minimal memory required, facilitating a fair evaluation of memory usage.

S1.5. Baseline training configuration

For our experiments, we utilized the unofficial implementations provided by Huggingface for Textual Inversion [4] and DreamBooth [14] without prior-preservation loss which is suitable to memory-constrained environment. For PEQA [5] and TuneQDM [15], we conducted experiments using reproduced code based on the Huggingface implementation of DreamBooth. QLoRA [2] was implemented using the `BitsandBytes` library. For Gradient-Free Textual Inversion [3], we utilized the official implementation for our experiments. We configured the training settings for each method to be as consistent as possible, adjusting only the batch size to 1. For DreamBooth, we set $\eta = 5 \times 10^{-6}$, $L = 400$, for Textual Inversion, we set $\eta = 5 \times 10^{-3}$, $L = 5,000$, for QLoRA, we set $\eta = 1 \times 10^{-4}$, $L = 500$, for PEQA, we set $\eta = 3 \times 10^{-6}$, $L = 400$, for TuneQDM, we set $\eta = 3 \times 10^{-6}$, $L = 400$, for Gradient-Free Textual Inversion, $\eta = 5 \times 10^{-6}$, $L = 13,000$, intrinsic dimension $d_i = 256$, $\sigma = 1$, and $\alpha = 1$.

S2. Additional Ablation Study

We conducted an extensive ablation study to analyze the impact of each component of ZOODiP. Beyond evaluating RGE, we examined the effectiveness of alternative gradient estimation methods, the influence of varying perturbation scales on performance, the impact of changing the number of gradient estimations, and the resulting time overhead associated with these changes. Unless otherwise noted, all additional experiments were performed on two subjects: `<dog6>` as a representative of living objects and `<shiny_sneaker>` as a representative of non-living objects. The results from these experiments were averaged to ensure a balanced and comprehensive evaluation.

μ	CLIP-T↑	CLIP-I↑	DINO↑
1×10^{-2}	0.281	0.778	0.599
$\mathbf{1 \times 10^{-3}}$	0.277	**0.797**	**0.613**
1×10^{-4}	**0.296**	0.724	0.470

Table S5. Quantitative results for different μ values. Optimal performance is observed at $\mu = 10^{-3}$ with varying μ, and we have set the value of μ accordingly for ZOODiP.

S2.1. Gradient estimation method

There are various methods for estimating gradients, each with unique advantages and trade-offs. In ZOODiP, we opted for Random Gradient Estimation (RGE, Eq. S1) due to its efficiency in estimating gradients using Monte Carlo gradient estimation. While many existing works utilize Simultaneous Perturbation Stochastic Approximation (SPSA, Eq. S2), which perturbs the gradient in two directions, or a one-point estimation method (Eq. S3) that calculates the gradient directly at the perturbed point, we evaluated all three methods for gradient estimation as seen in Tab. S4.

$$\hat{g}_\theta = \frac{1}{n}\sum_{i=1}^{n}\left[\frac{\mathcal{L}_{\text{LDM}}(\theta + \mu e_i) - \mathcal{L}_{\text{LDM}}(\theta)}{\mu} e_i\right] \tag{S1}$$

$$\hat{g}_\theta = \frac{1}{n}\sum_{i=1}^{n}\left[\frac{\mathcal{L}_{\text{LDM}}(\theta + \mu e_i) - \mathcal{L}_{\text{LDM}}(\theta - \mu e_i)}{2\mu} e_i\right] \tag{S2}$$

$$\hat{g}_\theta = \frac{1}{n}\sum_{i=1}^{n}\left[\frac{\mathcal{L}_{\text{LDM}}(\theta + \mu e_{i+1}) - \mathcal{L}_{\text{LDM}}(\theta + \mu e_i)}{\mu} e_i\right] \tag{S3}$$

Our results indicate that SPSA and RGE exhibit similar performance, with RGE occasionally outperforming SPSA. In contrast, the one-point estimation method performed poorly. From a computational perspective, SPSA requires $2n$ forward passes for n gradient estimation steps, whereas RGE needs only $n + 1$ forward passes, making it more efficient. Given its favorable balance of computational efficiency and performance, we selected RGE as the gradient estimation method for ZOODiP.

S2.2. Perturbation scale

We analyzed the impact of varying μ, the scaling parameter (*a.k.a* smoothing parameter) for perturbations in Eq. S1, on personalization performance. In general, the optimal value of μ can vary depending on the configuration of the model to be tuned as seen in Sec. S3.2 and Sec. S3.4. The results in Tab. S5 demonstrate that the chosen value of 10^{-3} is the most suitable perturbation scale for our setup.

n	CLIP-T↑	CLIP-I↑	DINO↑	Speed↑ (iter/s)
1	0.298	0.736	0.495	20.7
2	0.277	0.796	0.613	16.1
4	0.282	0.784	0.584	9.78
8	0.282	0.798	0.627	6.20

Table S6. Quantitative results for various n, the number of gradient estimations. $n = 2$ is the promising choice between performance and computation efficiency.

Method	PUTS	CLIP-T↑	CLIP-I↑	DINO↑
DB	✗	0.266	0.853	0.751
	✓	**0.272**	**0.854**	**0.758**
TI	✗	0.241	0.798	0.584
	✓	**0.247**	**0.825**	**0.661**

Table S7. Quantitative results from applying PUTS to DreamBooth and Textual Inversion. The results confirm that PUTS enhances the performance of gradient-based personalization methods by up to 13.2%. The improvement was particularly pronounced in Textual Inversion, which is highly influenced by the text encoder, highlighting the significant impact of PUTS in this context.

S2.3. Number of gradient estimation

According to MeZO [8], increasing n when fine-tuning large language models provides only marginal performance gains compared to the proportional increase in the number of forward passes, making $n = 1$ the most efficient choice. Tab. S6 illustrates the personalization performance across different n values with RGE. While increasing n slightly improves performance due to more accurate gradient estimation, the associated increase in training time becomes a limiting factor. Thus, we set $n = 2$ as a compromise between performance and efficiency.

S3. Additional Experiments

S3.1. PUTS on gradient-based methods

Partial Uniform Timestep Sampling (PUTS), one of the key components of ZOODiP, not only enhances ZOODiP's performance but also proves effective in gradient-based approaches. To validate this, we conducted experiments using two representative gradient-based personalization methods on Stable Diffusion v1.5: Textual Inversion and DreamBooth. These experiments were performed with the same hyperparameter settings for T_L and T_U as those used in ZOODiP. The results in Tab. S7 demonstrate that applying Partial Uniform Timestep Sampling (PUTS) to gradient-based methods improves performance by up to 11.6%. PUTS enhances efficiency by prioritizing informative timesteps, showcasing its versatility in optimizing both zeroth-order and gradient-based approaches.

U-Net Precision	CLIP-T↑	CLIP-I↑	DINO↑
INT8	**0.288**	0.834	**0.657**
INT4	0.212	**0.835**	0.647

Table S8. The performance comparison between INT8 and INT4 for U-Net precision on the `<dog6>` subset of the DB dataset.

Figure S1. Generated images with the prompt *"a photo of a dog"* with various weight precision. While INT8 precision produces results nearly equivalent to full-precision performance, INT4 precision exhibits noticeable degradation in image quality, highlighting the trade-off between lower precision and fidelity.

S3.2. 4-bit quantized models

We applied ZOODiP to a 4-bit quantized model of the U-Net, the component with the highest VRAM usage. We observed that quantizing with INT4 precision led to a performance loss in the quantization library we utilized as seen in Fig. S1. Consequently, we also noted a degradation in the performance of the personalized results when using this quantization approach. Nevertheless, our results demonstrate that ZOODiP is fully capable of personalizing 4-bit quantized diffusion models using only **1.9 GB** of memory during the entire training process. Fig.S2 and Tab.S8 present the qualitative and quantitative results, respectively, for personalization with the 4-bit quantized model. To utilize the `qint4` data type, we employed BFloat16 (BF16) for activation in all units, including the VAE, U-Net, and text encoder. When using BF16, the default μ value of 10^{-3} was insufficient, so we adjusted it to 10^{-2} to achieve optimal performance.

S3.3. Diversity of generated images

TI-based methods tend to exhibit less overfitting compared to DreamBooth (DB)-based methods, resulting in higher diversity in the generated images. To evaluate this tendency, we introduce the $\text{DINO}_{\text{inter}}$ score. For each subject, we generated 50 images using the same prompt with a personalized model. The DINOv2 embedding cosine similarity was then calculated for all pairs of generated images and averaged. A higher $\text{DINO}_{\text{inter}}$ score indicates that the generated images are more similar to each other, reflecting lower diversity in

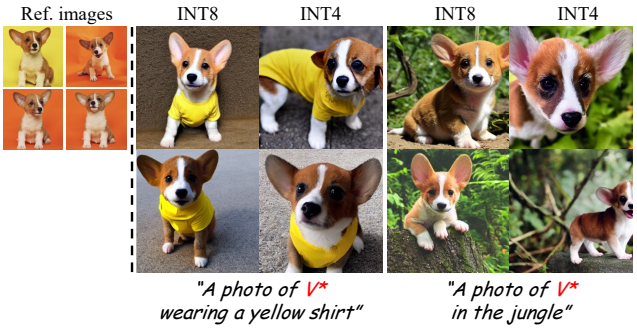

Figure S2. Qualitative results of U-Net precision at INT8 and INT4 in <dog6> dataset. ZOODiP works on INT4 and INT8, but performance diminishes due to degradation caused by INT4 quantization.

Method	$\text{DINO}_{\text{inter}} \downarrow$
DB	0.825
QLoRA	0.731
PEQA	0.806
TuneQDM	0.778
TI	0.679
GF-TI	0.150
ZOODiP (Ours)	0.671

Table S9. Comparison of $\text{DINO}_{\text{inter}}$ scores across various personalization methods. The results indicate that TI-based methods are capable of generating a diverse range of images, whereas DB-based methods exhibit relatively lower diversity. This observation was consistent across all subjects in the DreamBooth (DB) dataset, highlighting a fundamental difference in the ability of these methods to capture and reflect variations in the generated outputs.

the generated outputs. The quantitative results for $\text{DINO}_{\text{inter}}$ are presented in the Tab. S9. Additionally, we provide qualitative diversity results in Fig. S3. Fig. S3 shows a comparison of diversity across different methods, clearly showing that TI-based methods achieve significantly higher diversity than other approaches.

S3.4. Generalizability to other models

In the main paper, our experiments were conducted using Stable Diffusion v1.5 (SD1.5). However, ZOODiP is not limited to the certain model and can be extended to other models trained on different datasets and architectures, such as SD2.1 and SDXL [11] as seen in Fig. S4. For SD2.1, training was performed on a larger text token embedding dimension (1024 in `OpenCLIP-ViT-H`) using the same hyperparameters as SD1.5 except for $\mu = 10^{-2}$ and $\eta = 10^{-2}$. For SDXL, we maintained all hyperparameters except for $\mu = 10^{-2}$, $T_L = 700$, $L = 20,000$ and trained both token embeddings of `OpenCLIP-ViT-L` (768 dimension) and `OpenCLIP-ViT-G` (1280 dimension). The adjustment of T_L is supported by prior research which indicates that information loss vary with dimensionality [1, 7], and the value of T_L was selected empirically. Notably, full-precision (FP32) Textual Inversion on SDXL requires approximately 17 GB of memory, and DreamBooth cannot be trained on customer level GPUs, such as the RTX3090. In contrast, tuning SDXL with ZOODiP enables successful training of concepts with only **5 GB of memory**, highlighting its efficiency and scalability across diverse model configurations.

Base.	Method	Time (min)
DB	DB	2
	QLoRA	1.1
	PEQA	1.5
	TuneQDM	1.4
TI	TI	8.8
	GF-TI	293
	ZOODiP ($n = 2$)	31

Table S10. Total training time with the configurations in Sec S1.5. DB-based methods consume more memory but train faster, while TI-based methods are more memory-efficient but require more iterations.

S3.5. Training time

We measured the total training time for each personalization method and observed a general inverse relationship between memory usage and training time in Tab. S10. The table shows a trade-off between time complexity and space complexity. However, Gradient-Free Textual Inversion (GF-TI) deviates from this trend, requiring significantly more training time despite its low memory usage. This behavior indicates that GF-TI struggles with learning effectively under small batch training conditions, which impacts its overall efficiency and practicality in resource-constrained devices.

S4. Additional Results

S4.1. Qualitative results of SG and PUTS

Subspace Gradient (SG) and Partial Uniform Timestep Sampling (PUTS) play crucial roles in enabling efficient personalization by addressing two key challenges: SG removes noisy gradient components to focus on the most informative subspace, while PUTS suppresses sampling from ineffective timesteps to optimize training efficiency. To evaluate the impact of these components during the learning process, we generated images using the same seed with token embeddings learned at every $1,000$ iterations. As illustrated in the Fig. S5, using both SG and PUTS significantly accelerate the training process compared to naïve zeroth-

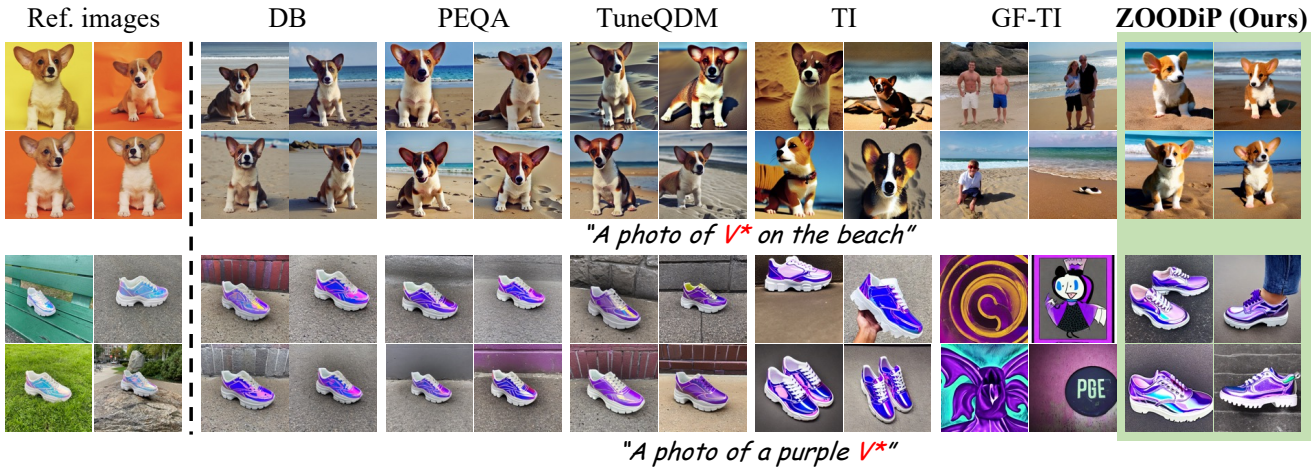

Figure S3. **Qualitative comparison of the diversity of generated images** This figure compares the diversity achieved by different personalization methods. ZOODiP demonstrates the ability to generate highly diverse images while utilizing minimal memory resources.

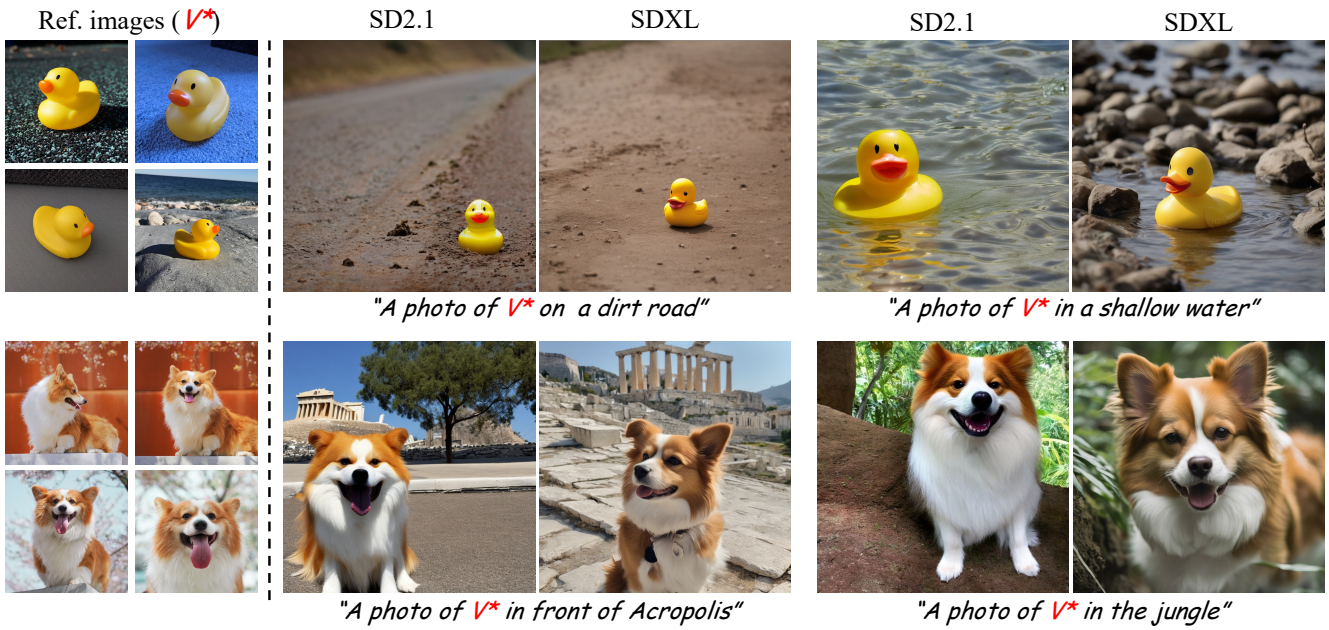

Figure S4. Qualitative results for personalizing SD2.1 and SDXL with ZOODiP. The figure demonstrate that ZOODiP can be applied not only to SD1.5, as discussed in the main paper, but also to various other models. For SD2.1, inference were conducted with images at a resolution of 768×768, while for SDXL, image generation was performed with resolution of 1024×1024. However, for SDXL, it was observed that the model's inherent color interpretation prevents the subject's colors from being completely replicated. This indicates that the model's color rendering can vary depending on the environmental context, leading to shifts in the perceived color scheme.

order optimization that does not incorporate these enhancements. The results show that models using SG and PUTS converge to high-quality personalization much faster than the baseline, underscoring the effectiveness of these components in improving the efficiency and speed of training.

S4.2. Additional qualitative results

In this section, we present additional qualitative comparisons of personalization results that were not included in the main paper. These results are illustrated in the Fig. S6, Fig. S7, Fig. S8, Fig. S9, and Fig. S10. The comparisons demonstrate that ZOODiP effectively captures the charac-

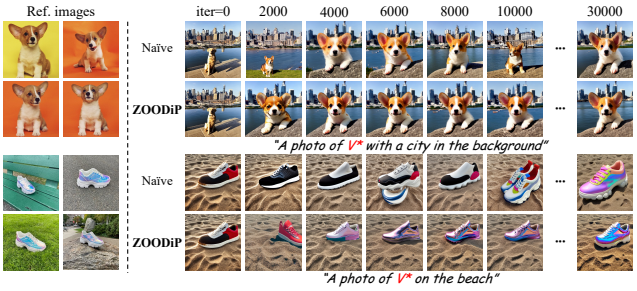

Figure S5. Qualitative comparisons on naïve ZO textual inversion without SG and PUTS to ZOODiP (Ours) over iterations. The naïve approach exhibits slower training and tends to produce images that are less aligned with the reference image. In contrast, ZOODiP achieves faster training and generates images that are closely aligned with the reference subject.

teristics of the reference image and text prompts, achieving a level of fidelity comparable to other gradient-based personalization methods.

S5. Limitation

In this section, we delve into the limitations of ZOODiP. While ZOODiP incorporates innovative techniques such as Subspace Gradient and Partial Uniform Timestep Sampling to mitigate the slow learning speed—a well-known challenge of zeroth-order optimization—it still requires a considerably larger number of iterations compared to gradient-based approaches. This increased iteration count stems from the fundamental nature of zeroth-order optimization, which relies on function evaluations rather than gradient backpropagation, inherently making it less sample-efficient. Although ZOODiP compensates with a faster training speed per iteration, the substantial number of iterations can introduce a significant time overhead, especially when proper hardware acceleration, such as high-performance Neural Processing Units (NPUs), is unavailable.

Furthermore, as ZOODiP is built upon the Textual Inversion framework, its performance is influenced by the strengths and weaknesses of Textual Inversion. This dependency implies that ZOODiP may face challenges in cases where Textual Inversion struggles, such as subjects with complex or highly variable visual characteristics, or when adapting to certain models that inherently perform poorly in personalization tasks. For example, if the base model lacks sufficient representational capacity or if the dataset used for training does not adequately capture the nuances of the subject, the effectiveness of ZOODiP can diminish.

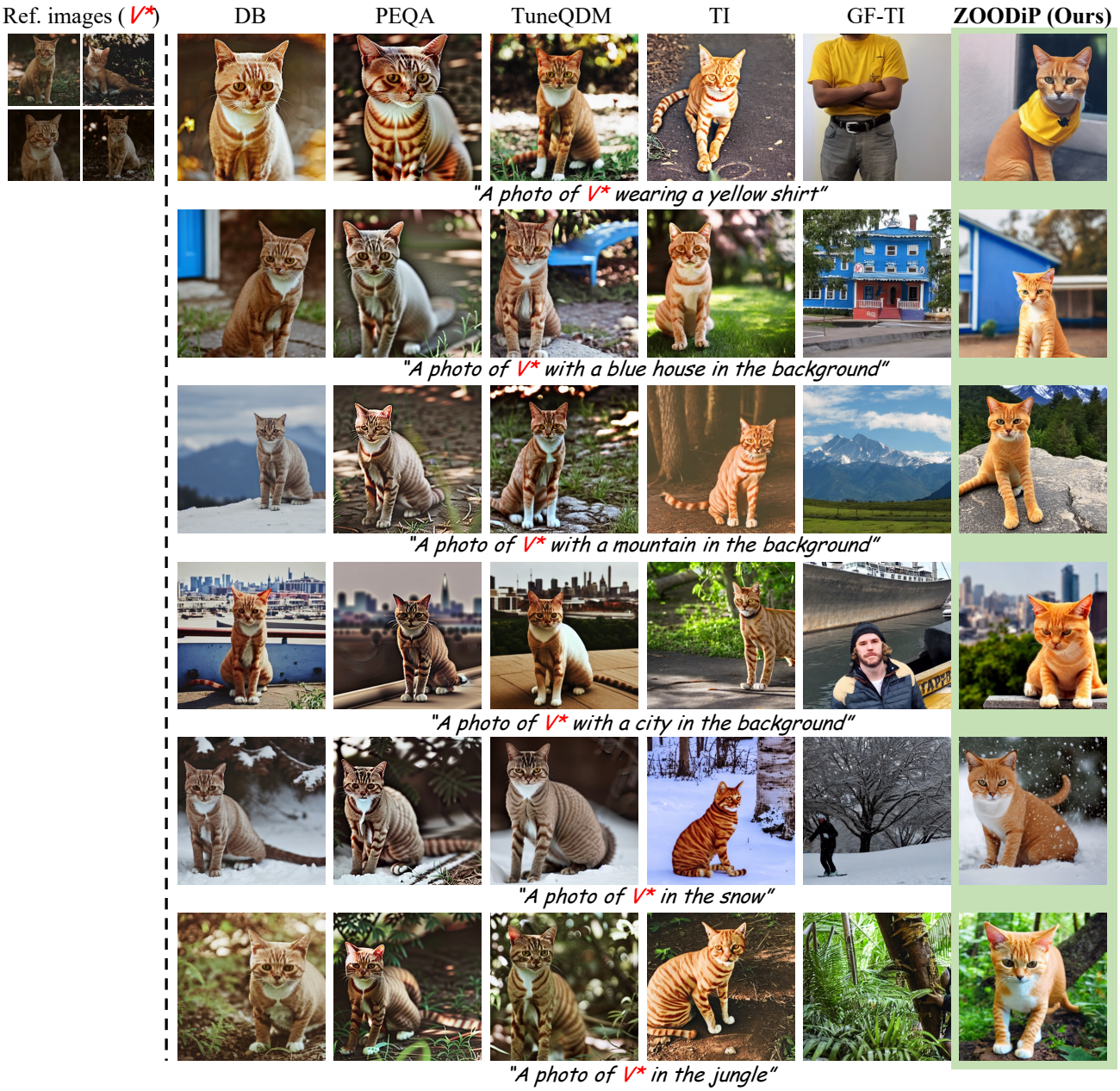

Figure S6. Qualitative comparison of image and text alignment on the <cat> subset of DB dataset.

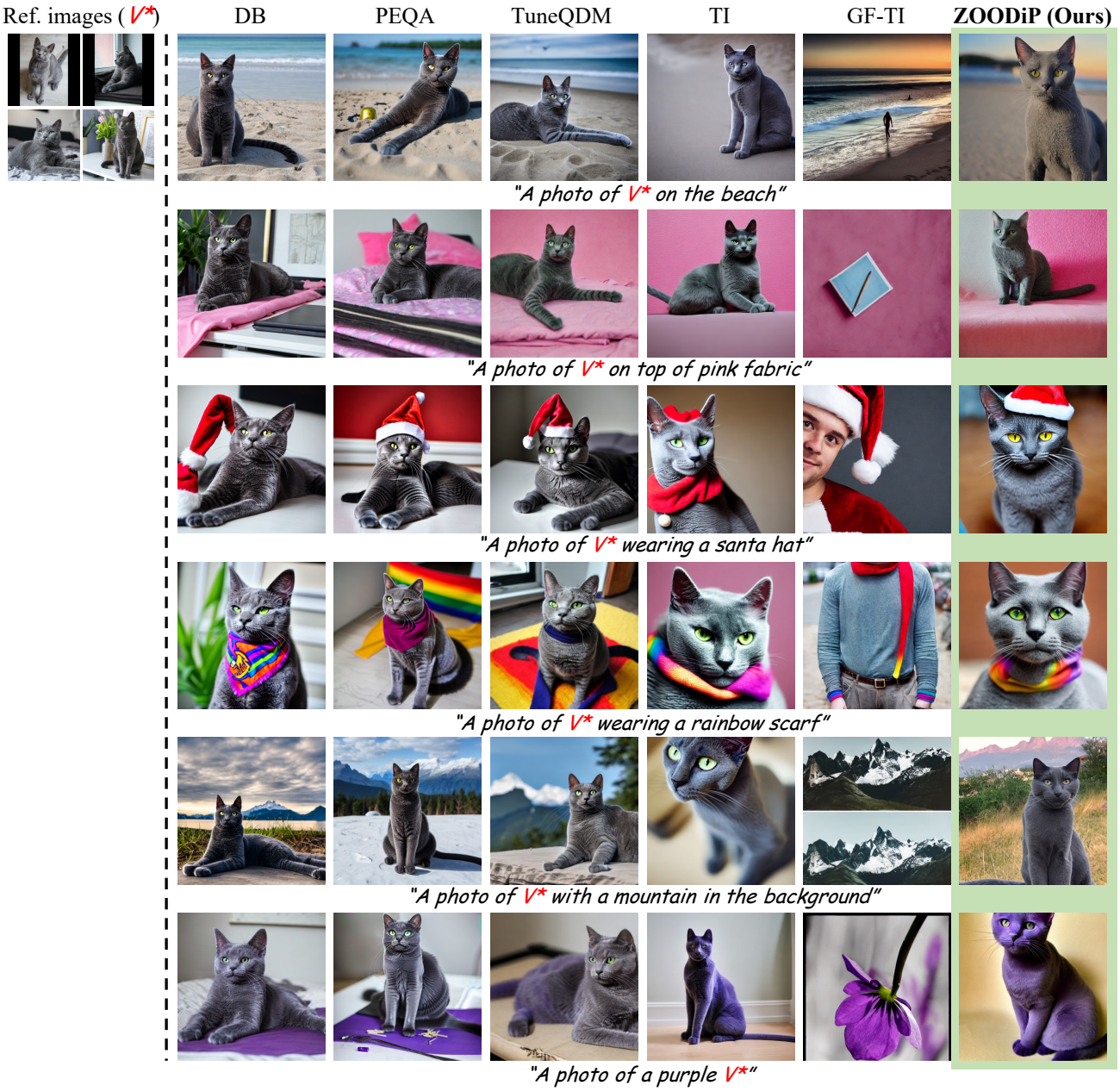

Figure S7. Qualitative comparison of image and text alignment on the <cat2> subset of DB dataset.

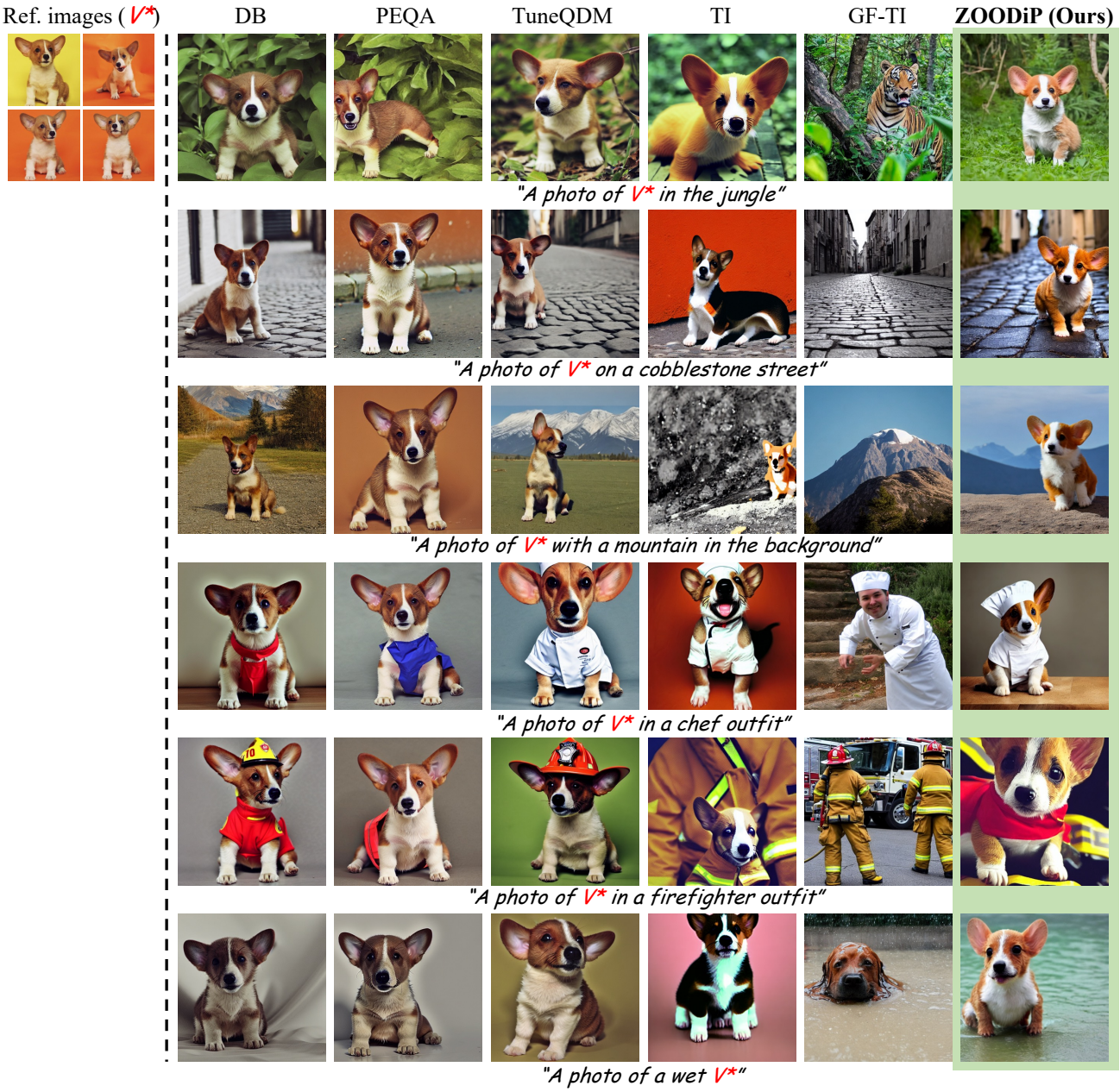

Figure S8. Qualitative comparison of image and text alignment on the <dog6> subset of DB dataset.

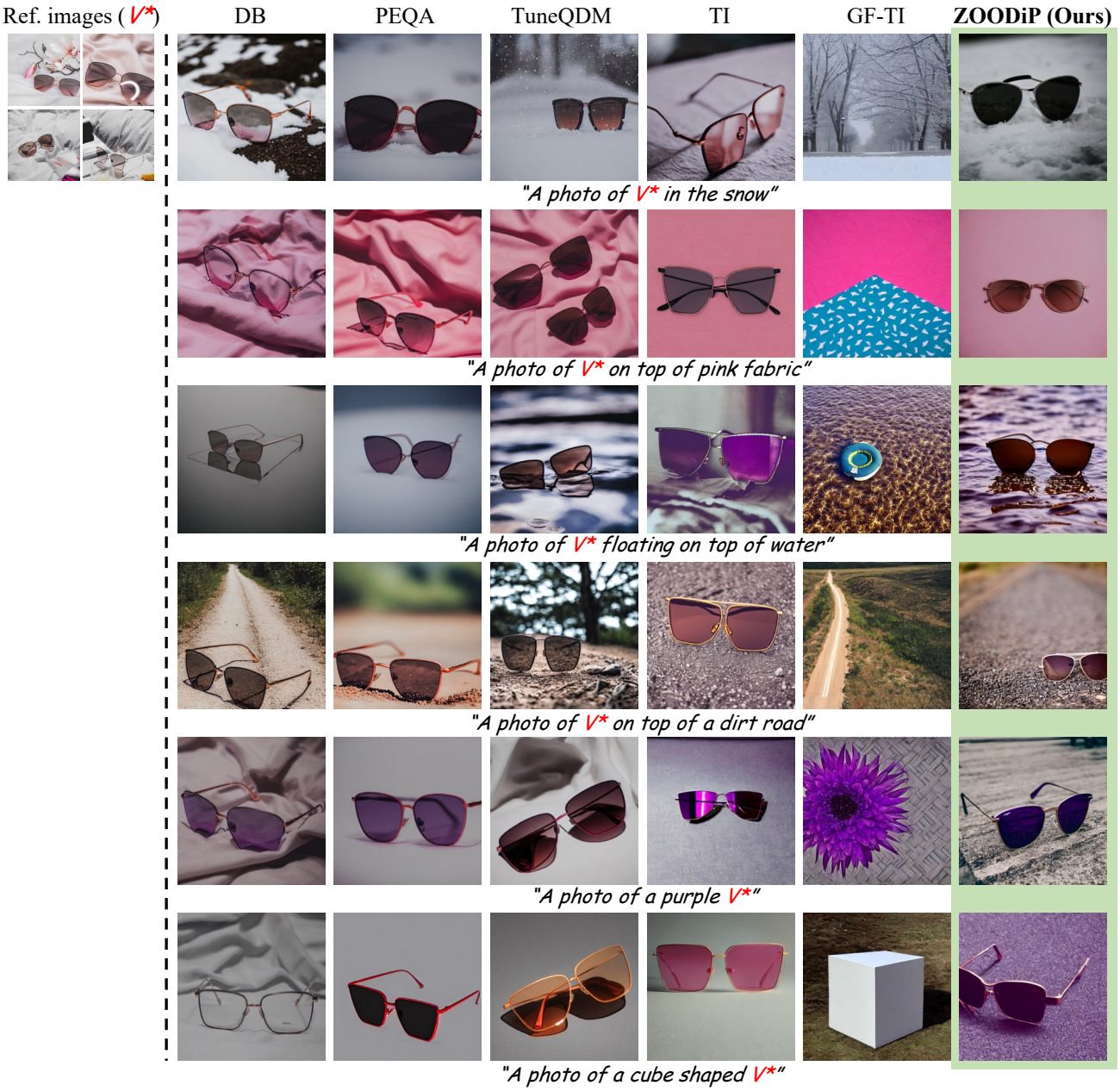

Figure S9. Qualitative comparison of image and text alignment on the <pink_sunglasses> subset of DB dataset.

Figure S10. Qualitative comparison of image and text alignment on the <shiny_sneaker> subset of DB dataset.

References

[1] CrossLabs. Diffusion with offset noise. Technical report, CrossLabs, 2023. 5

[2] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *NeurIPS*, 36, 2024. 3

[3] Zhengcong Fei, Mingyuan Fan, and Junshi Huang. Gradient-free textual inversion. In *ACM MM*, pages 1364–1373, 2023. 1, 3

[4] 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:2208.01618*, 2022. 1, 3

[5] Jeonghoon Kim, Jung Hyun Lee, Sungdong Kim, Joonsuk Park, Kang Min Yoo, Se Jung Kwon, and Dongsoo Lee. Memory-efficient fine-tuning of compressed large language models via sub-4-bit integer quantization. *NeurIPS*, 36, 2024. 3

[6] Kyungmin Lee, Sangkyung Kwak, Kihyuk Sohn, and Jinwoo Shin. Direct consistency optimization for compositional text-to-image personalization. *arXiv preprint arXiv:2402.12004*, 2024. 1

[7] Shanchuan Lin, Bingchen Liu, Jiashi Li, and Xiao Yang. Common diffusion noise schedules and sample steps are flawed. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 5404–5411, 2024. 5

[8] Sadhika Malladi, Tianyu Gao, Eshaan Nichani, Alex Damian, Jason D Lee, Danqi Chen, and Sanjeev Arora. Fine-tuning language models with just forward passes. *NeurIPS*, 36:53038–53075, 2023. 4

[9] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv:2304.07193*, 2023. 1

[10] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *NeurIPS*, 32, 2019. 1

[11] 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:2307.01952*, 2023. 5

[12] 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 *ICML*, pages 8748–8763. PMLR, 2021. 1

[13] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, pages 10684–10695, 2022. 1

[14] 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*, pages 22500–22510, 2023. 1, 2, 3

[15] Hyogon Ryu, Seohyun Lim, and Hyunjung Shim. Memory-efficient fine-tuning for quantized diffusion model. *arXiv:2401.04339*, 2024. 3