DiffuseKronA: A Parameter Efficient Fine-tuning Method for Personalized Diffusion Models

Supplementary Material

Project Page: https://diffusekrona.github.io/

Table of Contents

1. Background

Primarily in 1998, the practical implications of the Kronecker product were introduced in [17] for the task of image restoration. This study presented a flexible preconditioning approach based on Kronecker product and singular value decomposition (SVD) approximations. The approach can be used with a variety of boundary conditions, depending on what is most appropriate for the specific deblurring application. In the realm of parameter-efficient fine-tuning (PEFT) of large-scale models in deep learning, several literature studies [4, 11, 24, 25] have explored the efficacy of Kronecker products, illustrating their applications across diverse domains.

In context, COMPACTER [6] was the first line of work that proposes a method for fine-tuning large-scale language models with a better trade-off between task performance and the number of trainable parameters than prior work. It builds on top of ideas from adapters [12], low-rank optimization [14] (by leveraging Kronecker products), and parameterized hypercomplex multiplication layers [28]. KroneckerBERT [24] significantly compressed Pre-trained Language Models (PLMs) through Kronecker decomposition and knowledge distillation. It leveraged Kronecker decomposition to compress the embedding layer and the linear mappings in the multi-head attention, and the feed-forward network modules in the Transformer layers within BERT [3] model. The model outperforms state-of-the-art compression methods on the GLUE and SQuAD benchmarks. In a similar line of work, KronA [4] proposed a Kronecker product-based adapter module for efficient fine-tuning of Transformer-based PLMs (T5 [21]) methods on the GLUE benchmark.

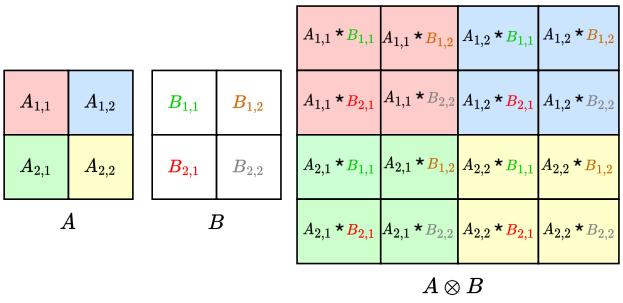


Figure 1. Demonstrating the functioning of the Kronecker product.

Apart from the efficient fine-tuning of PLMs, studies also shed some light on applying Kronecher products in the compression of convolution neural networks (CNNs) and vision transformers (ViTs). For instance, in [8], the authors compressed CNNs through generalized Kronecker product decomposition (GKPD) with a fundamental objective to reduce both memory usage and the required floating-point operations for convolutional layers in CNNs. This approach offers a plug-and-play module that can be effortlessly incorporated as a substitute for any convolutional layer, offering a convenient and adaptable solution. Recently proposed, KAdaptation [11] studies parameter-efficient model adapta-

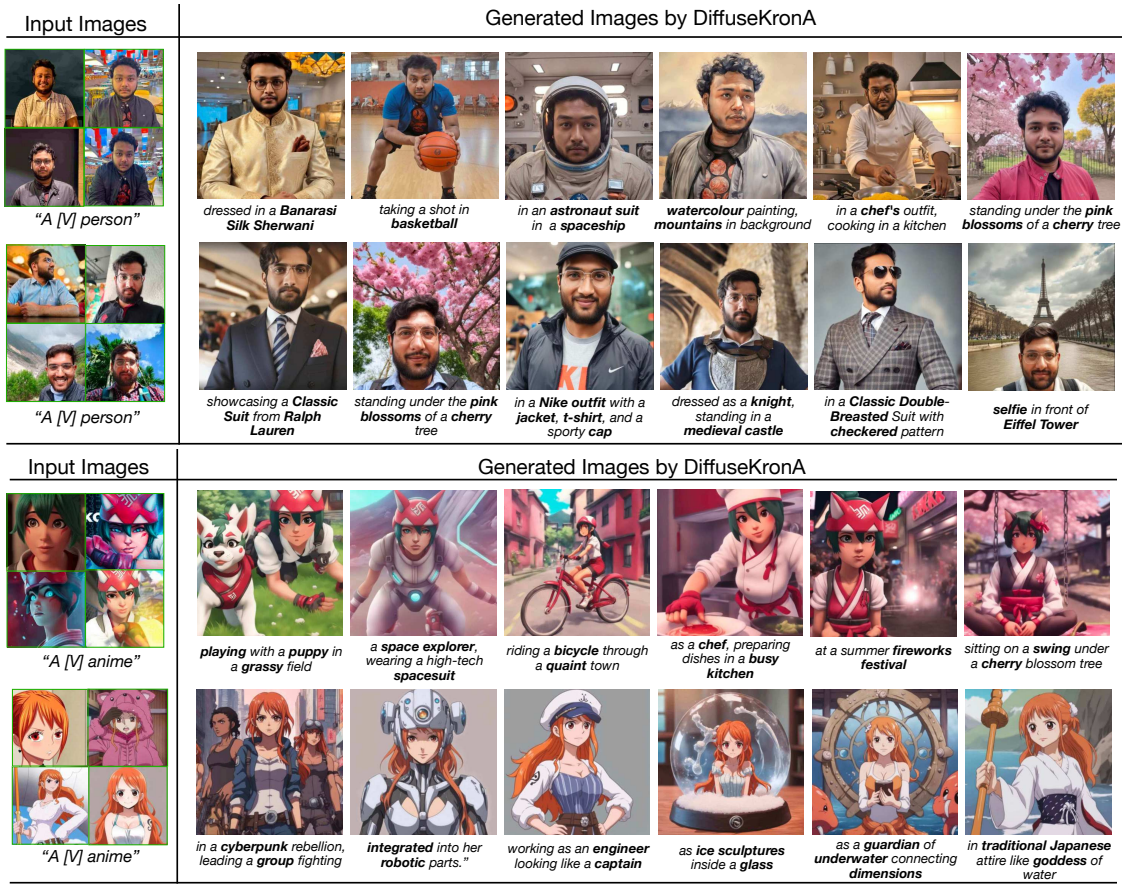


Figure 2. The results for the human face and anime characters generation highlight our method's endless application in creating portraits, animes, and avatars.

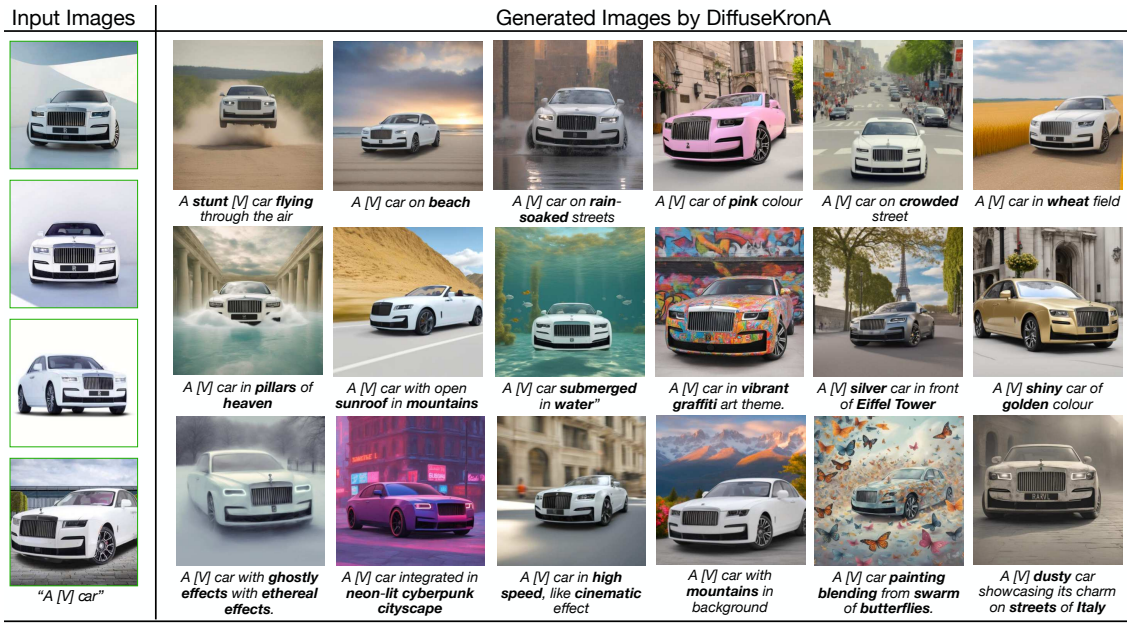


Figure 3. Results for car modifications and showcasing our method's potential application in the Automobile industry.

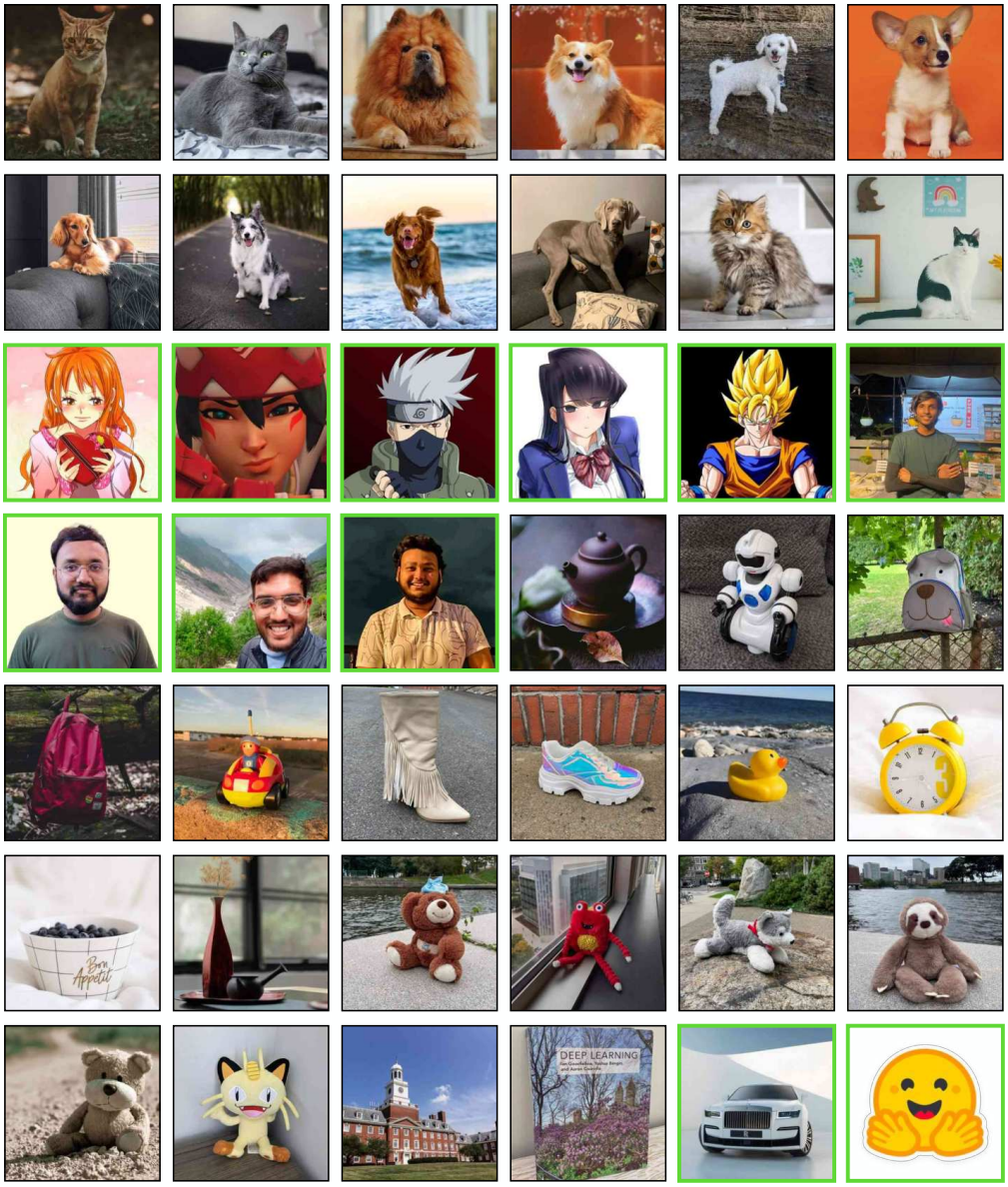


Figure 4. A collection of sample images representing all individual subjects involved in this study. Our collected subjects are highlighted in green.

tion strategies for ViTs on the image classification task. It formulates efficient model adaptation as a subspace training problem via Kronecker Adaptation (KAdaptation) and performs a comprehensive benchmarking over different efficient adaptation methods.

On the other hand, authors of [25] compressed RNNs for resource-constrained environments (e.g. IOT devices) using Kronecker product (KP) by 15-38x with minimal accuracy loss and by quantizing the resulting models to 8 bits, the compression factor is further pushed to 50x. In [26], RNNs are compressed based on a novel Kronecker CANDECOMP/PARAFAC (KCP) decomposition, derived from Kronecker tensor (KT) decomposition, by proposing two fast algorithms of multiplication between the input and the tensor-decomposed weight.

Besides all of the above, Kronecker decomposition is also being applied for GPT compression [5] which attempts to compress the linear mappings within the GPT-2 model. The proposed model, Kronecker GPT-2 (KnGPT2) is initialized based on the Kronecker decomposed version of the GPT-2 model. Subsequently, it undergoes a very light pre-training on only a small portion of the training data with intermediate layer knowledge distillation (ILKD).

From the aforementioned literature study, we have witnessed

the efficacy of Kronecker products for the task of model compression within various domains including NLP, RNN, CNN, ViT, and GPT space. Consequently, it has sparked considerable interest in exploring its impact on Generative models.

2. Datasets Descriptions

We have incorporated a total of 25 datasets from DreamBooth [22], encompassing images of backpacks, dogs, cats, and stuffed animals. Additionally, we integrated 7 datasets from custom diffusion [13] to introduce variety in our experimentation. To assess our model's ability to capture spatial features on faces, we curated a dataset consisting of 4 to 7 images each of 4 humans, captured from different angles. To further challenge our model against complex input images and text prompts, we compiled a dataset featuring 6 anime images from various sources. All datasets are categorized into four groups: ***living animals***, ***non-living objects***, ***anime***, and ***human faces***. Furthermore, the keywords utilized for fine-tuning the model remain consistent with those specified in the original papers. In Fig. 4, we present a sample image for all the considered subjects used in this study.

Image Attribution. Our collected datasets are taken from the following resources:

- **Rolls Royce:**
  - https://www.peakpx.com/en/hd-wallpaper-desktop-pxxec
  - https://4kwallpapers.com/cars/rolls-royce-ghost-2020/white-background-5k-8k-2554.html
  - https://www.cardekho.com/Rolls-Royce/Rolls-Royce_Ghost/pictures#leadForm
  - https://www.rolls-roycemotorcars.com/en_US/showroom/ghost-digital-brochure.html
- **Hugging Face:** https://huggingface.co/brand
- **Nami:**
  - http://m.gettywallpapers.com/nami-pfps-2/
  - https://tensor.art/models/616615209278282245
  - https://www.facebook.com/NamiHotandCute/?locale=bs_BA
  - https://k.sina.cn/article_1655152542_p62a79f9e02700nhhe.html
- **Kiriko:**
  - https://in.pinterest.com/pin/306948530865002366/
  - https://encrypted-tbn2.gstatic.com/images?q=tbn:ANd9GcSDSk98Uw3O2XW_RFC1jD_Kmw70JWU459euVYtU9nn1CpzPDcwS
  - https://comisc.theothertentacle.com/overwatch+kiriko+fanart
  - https://www.1999.co.jp/eng/11030018
- **Shoko Komi:**
  - https://wallpaperforu.com/tag/komi-shouko-wallpaper/page/2/
  - https://www.tiktok.com/@anime_geek00/video/7304798157894995243
  - https://wall.alphacoders.com/big.php?i=1305702
  - http://m.gettywallpapers.com/komi-can-t-communicate-wallpapers/
- **Kakashi Hatake:**
  - https://www.ranker.com/list/best-kakashi-hatake-quotes/ranker-anime?page=2;
  - https://www.wallpaperflare.com/search?wallpaper=Hatake+Kakashi
  - https://www.peakpx.com/en/hd-wallpaper-desktop-kiptm
  - https://in.pinterest.com/pin/584060645404620659/

3. Evaluation Metrics

We utilize metrics introduced in DreamBooth [22] for evaluation: DINO and CLIP-I scores measure subject fidelity, while CLIP-T assesses image-text alignment. The DINO score is the normalized pairwise cosine similarity between the `ViT-S/16 DINO` embeddings of the generated and input (real) images. Similarly, the CLIP-I score is the normalized pairwise `CLIP ViT-B/32 image` embeddings of the generated and input images. Meanwhile, the CLIP-T score computes the normalized cosine similarity between the given text prompt and generated image CLIP embeddings.

Let's denote the pre-trained CLIP Image encoder as $\mathcal{I}$, the CLIP text encoder as $\mathcal{T}$, and the DINO model as $\mathcal{V}$. We measure cosine similarity between two embeddings x and y as $sim(x,\ y) = \frac{x.y}{||x||\cdot||y||}$. Given two sets of images, we represent the input image set as $\mathcal{R} = \{R_i\}_{i=1}^{n}$ and generated image set as $\mathcal{G} = \{G_i\}_{i=1}^{m}$ corresponding to the prompt set $\mathcal{P} = \{P_i\}_{i=1}^{m}$, where m and n represents the number of generated and input images, respectively and $R, G \in \mathbb{R}^{3\times H\times W}$ (H and W is the height and width of the image). Then, CLIP-I image-to-image and CLIP-T text-to-image

Table 1. Average metrics (CLIP-I, CLIP-T, and DINO scores) from various prompt runs for each subject using our proposed method.

Subject	Cat	Cat2	Dog2	Dog	Dog3	Dog6
CLIP-I	0.858 ± 0.017	0.826 ± 0.030	0.833 ± 0.023	0.854 ± 0.015	0.789 ± 0.027	0.845 ± 0.031
CLIP-T	0.348 ± 0.033	0.343 ± 0.030	0.331 ± 0.028	0.349 ± 0.029	0.338 ± 0.025	0.323 ± 0.032
DINO	0.814 ± 0.025	0.752 ± 0.021	0.750 ± 0.049	0.856 ± 0.008	0.549 ± 0.060	0.788 ± 0.017
Subject	Dog5	Dog7	Dog8	Doggy	Cat3	Cat4
CLIP-I	0.824 ± 0.024	0.853 ± 0.015	0.829 ± 0.021	0.734 ± 0.031	0.834 ± 0.034	0.861 ± 0.016
CLIP-T	0.337 ± 0.026	0.334 ± 0.025	0.343 ± 0.026	0.329 ± 0.030	0.348 ± 0.029	0.349 ± 0.032
DINO	0.761 ± 0.001	0.730 ± 0.049	0.717 ± 0.050	0.686 ± 0.039	0.744 ± 0.031	0.863 ± 0.030
Subject	Nami (Anime)	Kiriko (Anime)	Kakshi (Anime)	Shoko Komi (Anime)	Harshit (Human)	Nityanand (Human)
CLIP-I	0.781 ± 0.035	0.738 ± 0.039	0.834 ± 0.028	0.761 ± 0.029	0.724 ± 0.018	0.665 ± 0.031
CLIP-T	0.337 ± 0.029	0.320 ± 0.032	0.318 ± 0.031	0.356 ± 0.028	0.297 ± 0.036	0.307 ± 0.030
DINO	0.655 ± 0.023	0.483 ± 0.041	0.617 ± 0.061	0.596 ± 0.024	0.555 ± 0.025	0.447 ± 0.068
Subject	Shyam (Human)	Teapot	Robot Toy	Backpack	Dog Backpack	Rc Car
CLIP-I	0.731 ± 0.015	0.836 ± 0.051	0.828 ± 0.026	0.907 ± 0.026	0.774 ± 0.037	0.797 ± 0.020
CLIP-T	0.297 ± 0.026	0.347 ± 0.025	0.285 ± 0.032	0.347 ± 0.021	0.333 ± 0.027	0.321 ± 0.027
DINO	0.531 ± 0.030	0.528 ± 0.132	0.642 ± 0.023	0.660 ± 0.088	0.649 ± 0.037	0.651 ± 0.065
Subject	Shiny Shoes	Duck	Clock	Vase	Plushie1	Monster Toy
CLIP-I	0.806 ± 0.025	0.845 ± 0.023	0.825 ± 0.062	0.827 ± 0.013	0.897 ± 0.014	0.782 ± 0.041
CLIP-T	0.308 ± 0.023	0.303 ± 0.016	0.308 ± 0.035	0.332 ± 0.026	0.308 ± 0.030	0.308 ± 0.029
DINO	0.735 ± 0.090	0.682 ± 0.049	0.590 ± 0.158	0.705 ± 0.025	0.813 ± 0.027	0.573 ± 0.060
Subject	Plushie2	Plushie3	Building	Book	Car	HuggingFace
CLIP-I	0.803 ± 0.022	0.792 ± 0.015	0.852 ± 0.013	0.695 ± 0.023	0.830 ± 0.024	0.810 ± 0.002
CLIP-T	0.324 ± 0.024	0.337 ± 0.031	0.268 ± 0.023	0.301 ± 0.022	0.299 ± 0.032	0.288 ± 0.042
DINO	0.728 ± 0.020	0.766 ± 0.033	0.742 ± 0.019	0.579 ± 0.040	0.684 ± 0.036	0.692 ± 0.001

similarity scores would be computed as S^I_{CLIP} and S^T_{CLIP}, respectively.

$$S^I_{CLIP} = \frac{1}{mn} \sum_{i=1}^{n} \sum_{j=1}^{m} sim(\mathcal{I}(R_i), \mathcal{I}(G_j)) \quad (1)$$

$$S^T_{CLIP} = \frac{1}{m} \sum_{i=1}^{m} sim(\mathcal{I}(G_i), \mathcal{T}(P_i)) \quad (2)$$

Similarly, the DINO image-to-image similarity score would be computed as

$$S_{DINO} = \frac{1}{mn} \sum_{i=1}^{n} \sum_{j=1}^{m} sim(\mathcal{V}(R_i), \mathcal{V}(G_j)). \quad (3)$$

Notably, the DINO score is preferred to assess subject fidelity owing to its sensitivity to differentiate between subjects within a given class. In personalized T2I generations, all three metrics should be considered jointly for evaluation to avoid a biased conclusion. For instance, models that copy training set images will have high DINO and CLIP-I scores but low CLIP-T scores, while a vanilla T2I generative model like SD and SDXL without subject knowledge will produce high CLIP-T scores with poor subject alignment. As a result, neither model is considered desirable for the subject-driven T2I generation. In Table-5, we showcase mean subject-specific CLIP-I, CLIP-T, and DINO scores along with standard deviations computed across 36 datasets, with a total of around 1600 generated images and prompts.

4. *DiffuseKronA* Ablations Study

As outlined in Sec. 4.2. of the main paper, we explore various trends and observations derived from extensive experimentation on the datasets specified in Fig. 4.

4.1. Choice of modules to fine-tune the model

Within the UNet network's transformer block, the linear layers consist of two components: a) attention matrices and b) a feed-forward network (FFN). Our investigation focuses on discerning the weight matrices with the highest importance for fine-tuning, aiming for efficiency in parameter utilization.

Our findings reveal that fine-tuning only the attention weight matrices, namely (W_K, W_Q, W_V, W_O), proves to be the most impactful and parameter-efficient strategy. Conversely, fine-tuning the FFN layers does not significantly enhance image synthesis quality but substantially increases the parameter count, approximately doubling the computational load. Refer to Fig. 5 for a visual representation

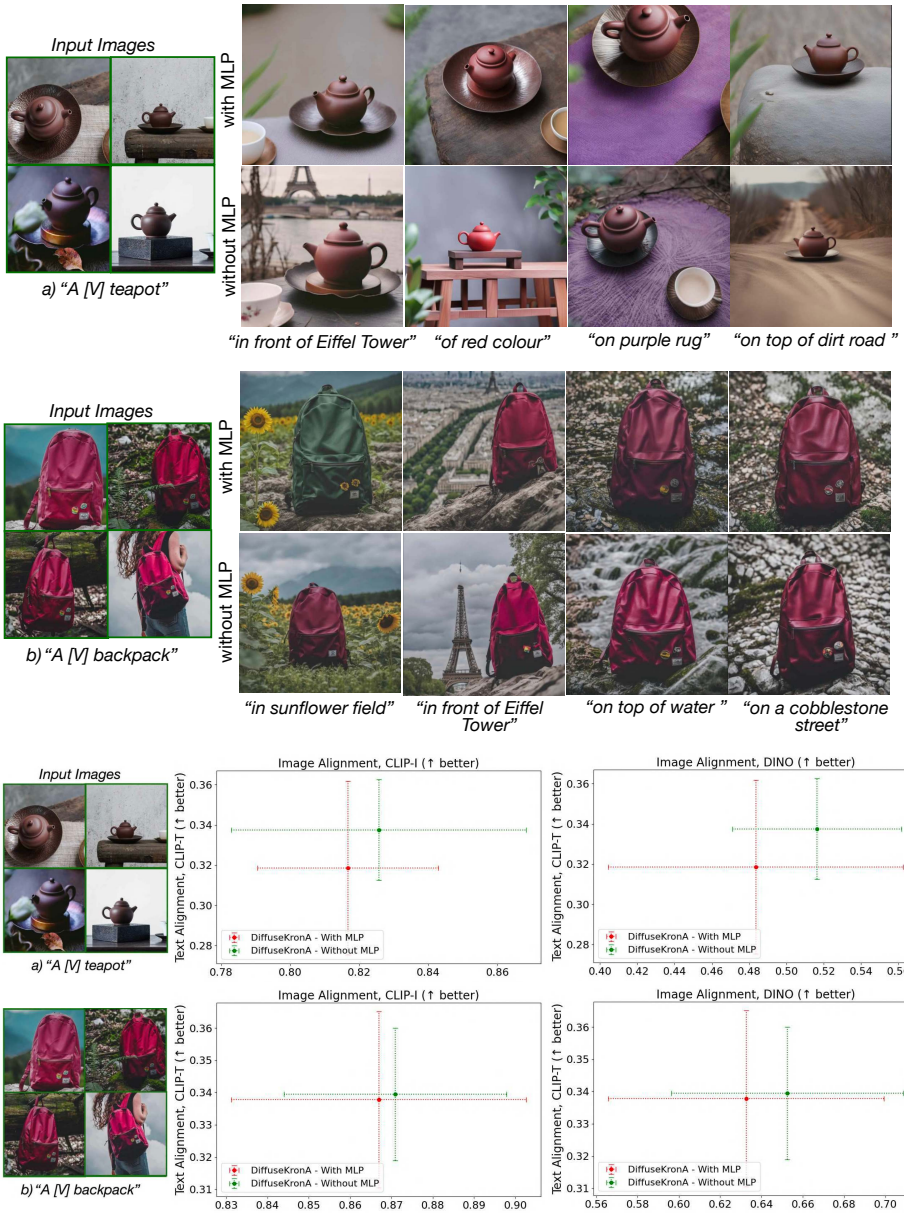


Figure 5. Qualitative and Quantitative comparison between fine-tuning with MLP and w/o MLP. Fine-tuning MLP layers introduces more parameters and doesn't enhance image generation compared with fine-tuning solely attention-weight matrices. So, the best outcomes and efficient use of parameters occur when only attention weight (without MLP) matrices are fine-tuned.

comparing synthesis image quality with and without fine-tuning FFN layers on top of attention matrices. This graph unequivocally demonstrates that incorporating MLP layers does not enhance fidelity in the results. On the contrary, it diminishes the quality of generated images in certain instances, such as *"A [V] backpack in sunflower field"*, while concurrently escalating the number of trainable parameters substantially, approximately 2x times.

This approach of exclusively fine-tuning attention layers not only maximizes efficiency but also helps maintain a lower overall parameter count. This is particularly advantageous when computational resources are limited, ensuring computational efficiency in the fine-tuning process.

4.2. Effect of Kronecker Factors

How to initialize the Kronecker factors? Initialization plays a crucial role in the fine-tuning process. Networks that are poorly initialized can prove challenging to train. Therefore, having a well-crafted initialization strategy is crucial for achieving effective fine-tuning. In our experiments, we explored three initialization methods: Normal initialization, Kaiming Uniform initialization [10], and Xavier initializa-

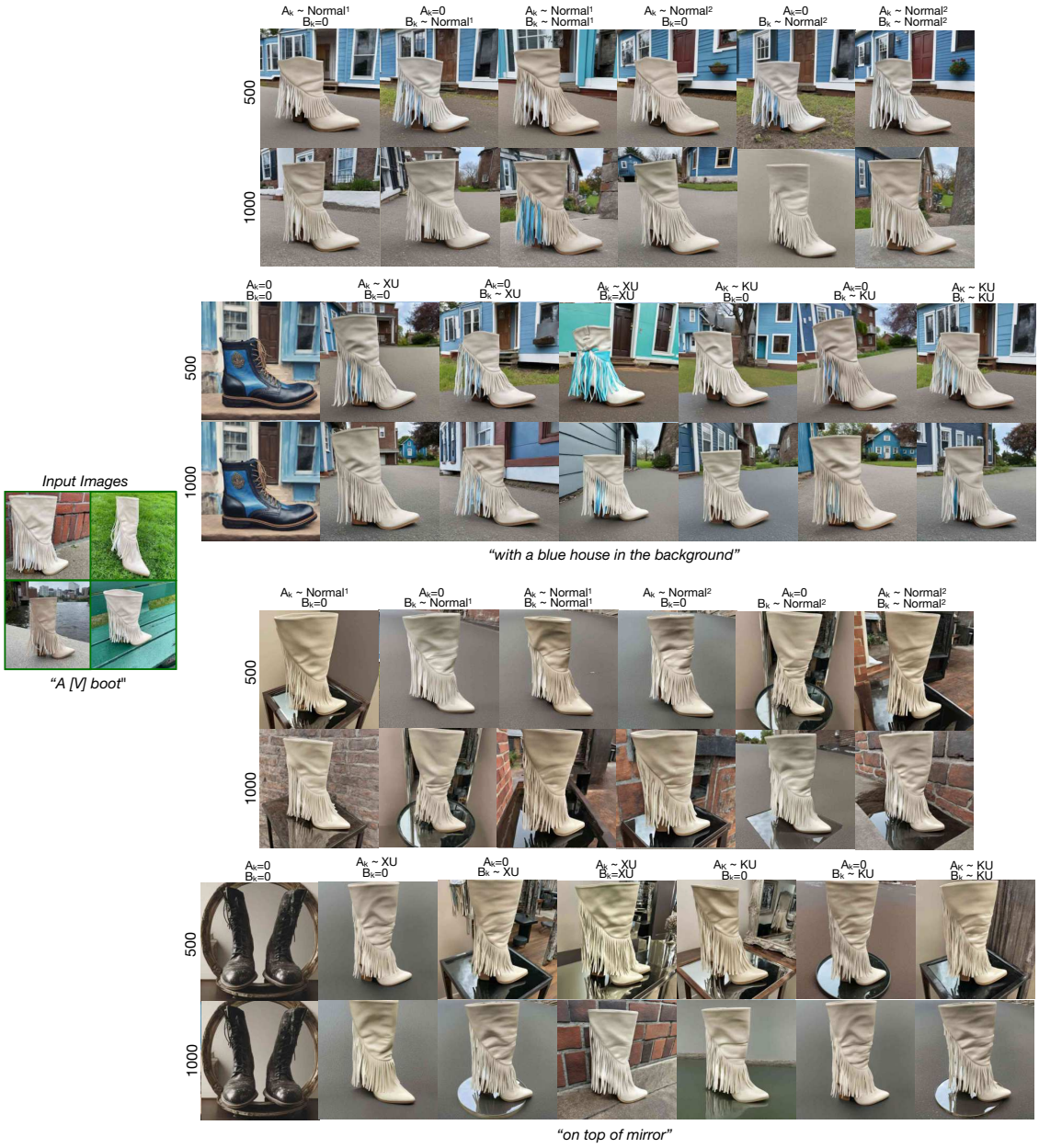


Figure 6. Impact of different initialization strategies: optimal outcomes are achieved when initializing B_k to zero while initializing A_k with either a Normal or Kaiming uniform distribution.

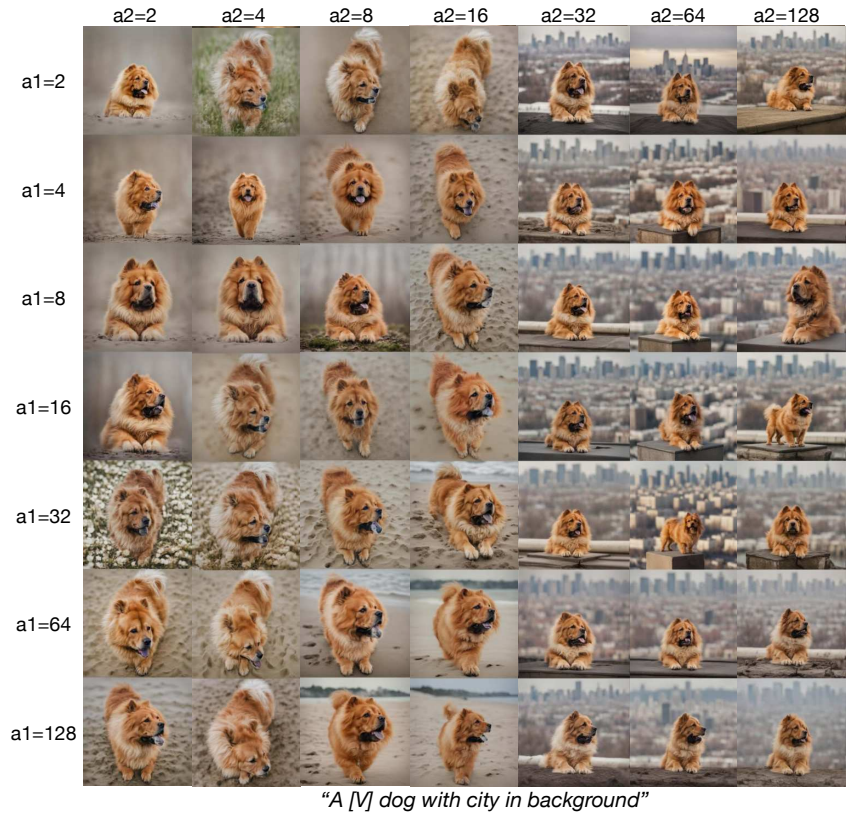
tion. These methods were applied to initialize the Kronecker factors A_k and B_k. We observed that initializing both factors with the same type of initialization failed to preserve fidelity. Surprisingly, initializing B_k with zero yielded the best results in the fine-tuning process.

As illustrated in Fig. 6, images initialized with (A_k = Normal^s, $B_k = 0$) and (A_k = KU, $B_k = 0$) produce the most favorable results, while images initialized with (A_k = Normal^s, B_k = Normal^s) and (A_k = XU, B_k = XU) result in the least satisfactory generations. Here, $s \in 1, 2$ denotes two different normal distributions - $\mathcal{N}(0, 1/a_2)$ and $\mathcal{N}\left(0, \sqrt{min(d,h)}\right)$ respectively, where d and h represents in features and out features dimension.

Effect of size of Kronecker Factors. The size of the Kronecker factors significantly influences the images generated by *DiffuseKronA*. Larger Kronecker factors tend to produce images with higher resolution and more detailing, while smaller Kronecker factors result in lower-resolution images with less detailing. Images generated with larger Kronecker factors tend to look more realistic, while those generated

Input Images

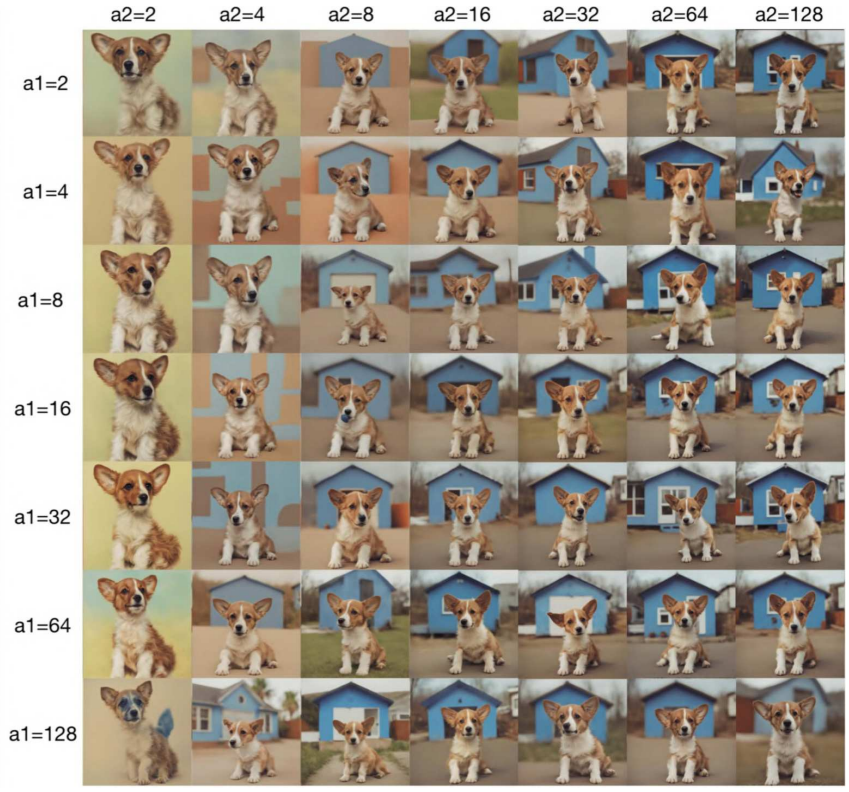
"A [V] dog"



"A [V] dog with city in background"

Input Images

"A [V] dog"



"A [V] dog with blue house in the background"

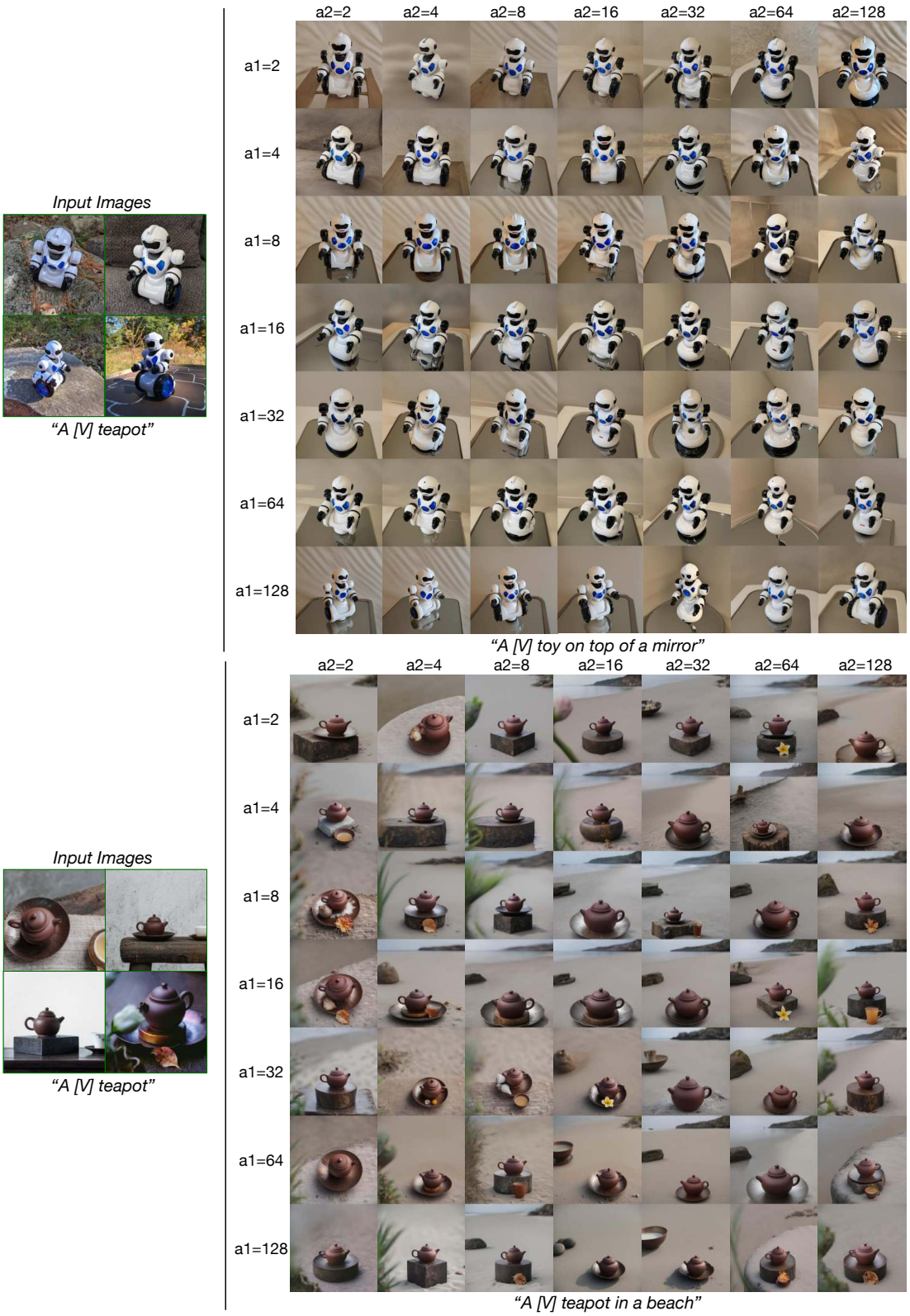


Figure 7. Effect of Kronecker factors *i.e.*, a_1 and a_2 in image generations. Optimal selection of a_1 and a_2 considers **image fidelity** and **parameter count**. Following this, we choose a_1 and a_2 as 4 and 64, respectively, interpreting that the lower Kronecker factor (A) should have a lower dimension compared to the upper Kronecker factor (B).

Table 2. Effect of the size of Kronecker factors (*i.e.* a_1 & a_2) in terms of trainable parameter count.

a_1	a_2	# Parameters
1	2	119399520
	4	59701440
	8	29854080
	16	14933760
	32	7480320
	64	3767040
	128	1937280

a_1	a_2	# Parameters
2	2	238799040
	4	119402880
	8	59708160
	16	29867520
	32	14960640
	64	7534080
	128	3874560

a_1	a_2	# Parameters
4	2	119402880
	4	59708160
	8	29867520
	16	14960640
	32	7534080
	64	3874560
	128	2152320

a_1	a_2	# Parameters
8	2	59708160
	4	29867520
	8	14960640
	16	7534080
	32	3874560
	64	2152320
	128	1506240

a_1	a_2	# Parameters
16	2	29867520
	4	14960640
	8	7534080
	16	3874560
	32	2152320
	64	1506240
	128	1613280

a_1	a_2	# Parameters
32	2	14960640
	4	7534080
	8	3874560
	16	2152320
	32	1506240
	64	1613280
	128	2526960

a_1	a_2	# Parameters
64	2	7534080
	4	3874560
	8	2152320
	16	1506240
	32	1613280
	64	2526960
	128	4704120

a_1	a_2	# Parameters
128	2	3874560
	4	2152320
	8	1506240
	16	1613280
	32	2526960
	64	4704120
	128	9233340

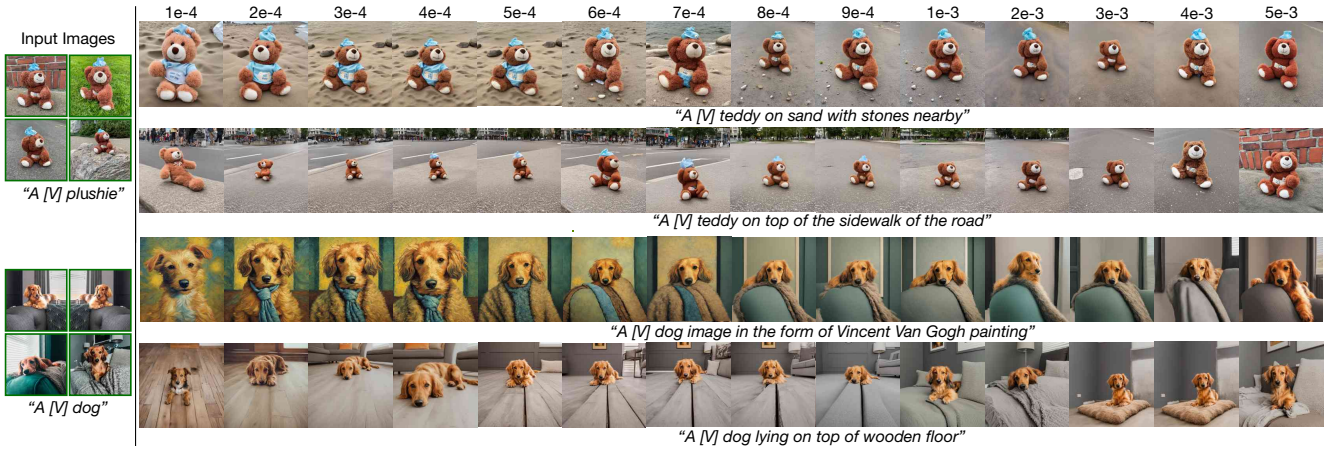


Figure 8. Effect of learning rate on subject fidelity and text adherence. The most favorable results are obtained using learning rate 5×10^{-4}.

with smaller Kronecker factors appear more abstract. Varying the Kronecker factors can result in a wide range of images, from highly detailed and realistic to more abstract and lower resolution.

In Fig. 7 when both a_1 and a_2 are set to relatively high values (8 and 64 respectively), the generated images are of very high fidelity and detail. The features of the dog and the house in the background are more defined and realistic with the house having a blue colour as mentioned in the prompt. When a_1 is halved (4) while maintaining the same (64), it results in images where the dog and the house are still quite detailed due to the high value of a_2, but perhaps less so than in the previous case due to the smaller value of a_1. However, when the factors are small ≤ 8, not only the generated images do not adhere to the prompt, but the number of trainable parameters increases drastically.

In Tab. 2, we present the count of trainable parameters corresponding to different Kronecker factors.

4.3. Effect of learning rate

The learning rate factor influences the alignment of generated images towards both text prompts and input images. Our approach yields better results when using learning rates near 5×10^{-4}. Higher learning rates, typically around 10^{-3}, compel the model to overfit, resulting in images closely mirroring the input images and largely ignoring the input text prompts. Conversely, lower learning rates, below 10^{-4}, cause the model to overlook the input images, concentrating solely on the provided input text.

In Fig. 8, for *"A [V] teddy on sand with stones nearby"* when the learning rate is $\geq 1 \times 10^{-3}$, the generated teddy bears closely resemble the input images. Additionally, the sand dunes in the images vanish, along with the removal of stones. Conversely, for learning rates in the intermediate ranges, the sand dunes and pebbles remain distinctly visible. In the context of *"A [V] dog image in the form of a Vincent Van Gogh painting"* in Fig. 8, images close to the

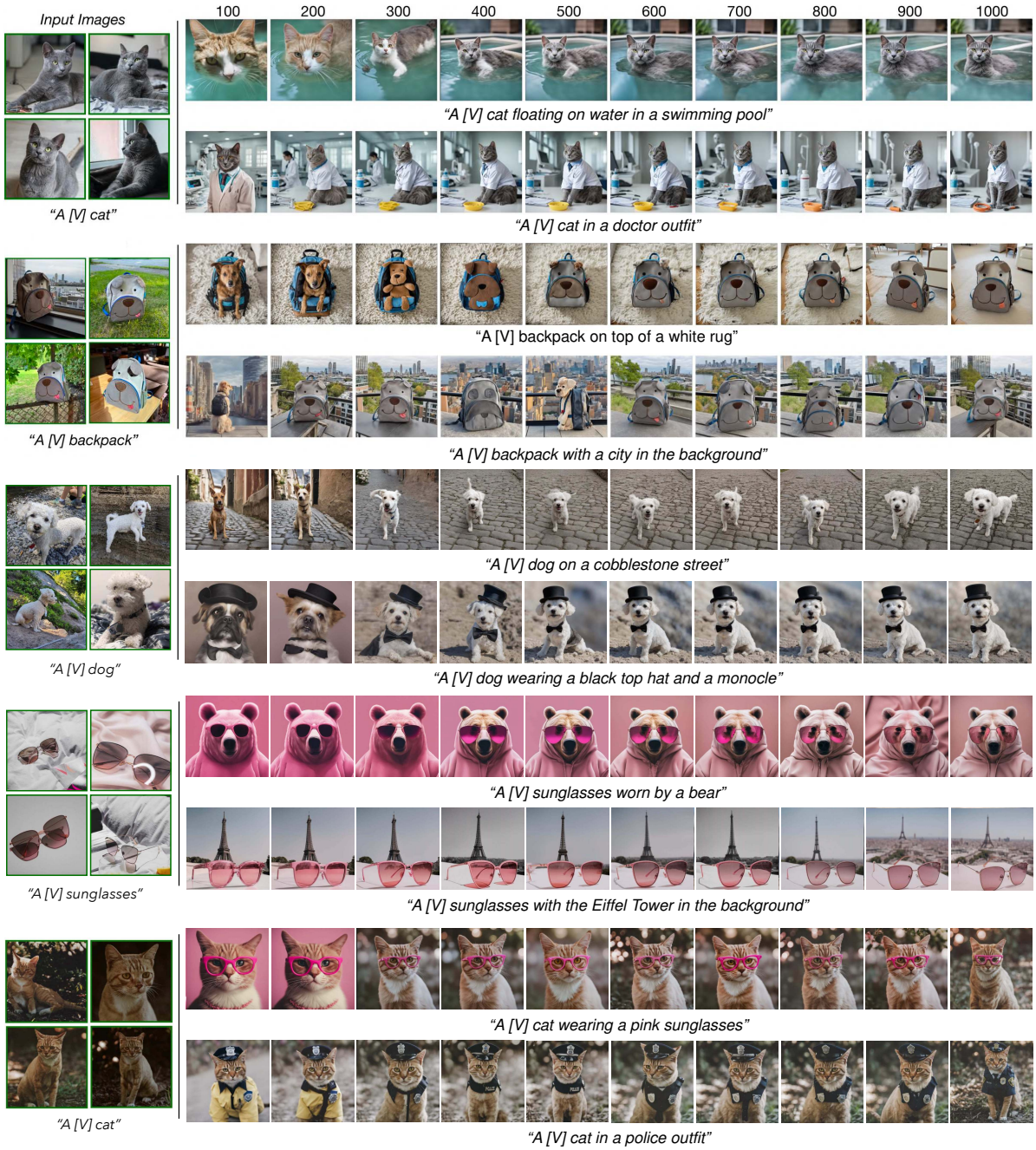


Figure 9. Effect of training steps in image generation on SDXL. In the case of simple prompts (row 1), *DiffuseKronA* consistently delivers favorable results between steps 500 and 1000. Conversely, for more complex prompts (row 2), reaching the desired outcome might necessitate waiting until after step 1000.

rightmost edge lack a discernible painting style, appearing too similar to the input images. Conversely, images near the leftmost side exhibit a complete sense of Van Gogh's style but lack the features present in the input images. Notably, in the images positioned in the middle, there is an excellent fusion of the painting style and the features of the input images.

4.4. Effect of training steps

In T2I personalization, the timely attainment of satisfactory results within a specific number of iterations is crucial. This not only reduces the overall training time but also helps prevent overfitting to the training images, ensuring efficiency and higher fidelity in image generation. With SDXL, we successfully generate desired-fidelity images within 500 iterations, if the input images and prompt complexity are not

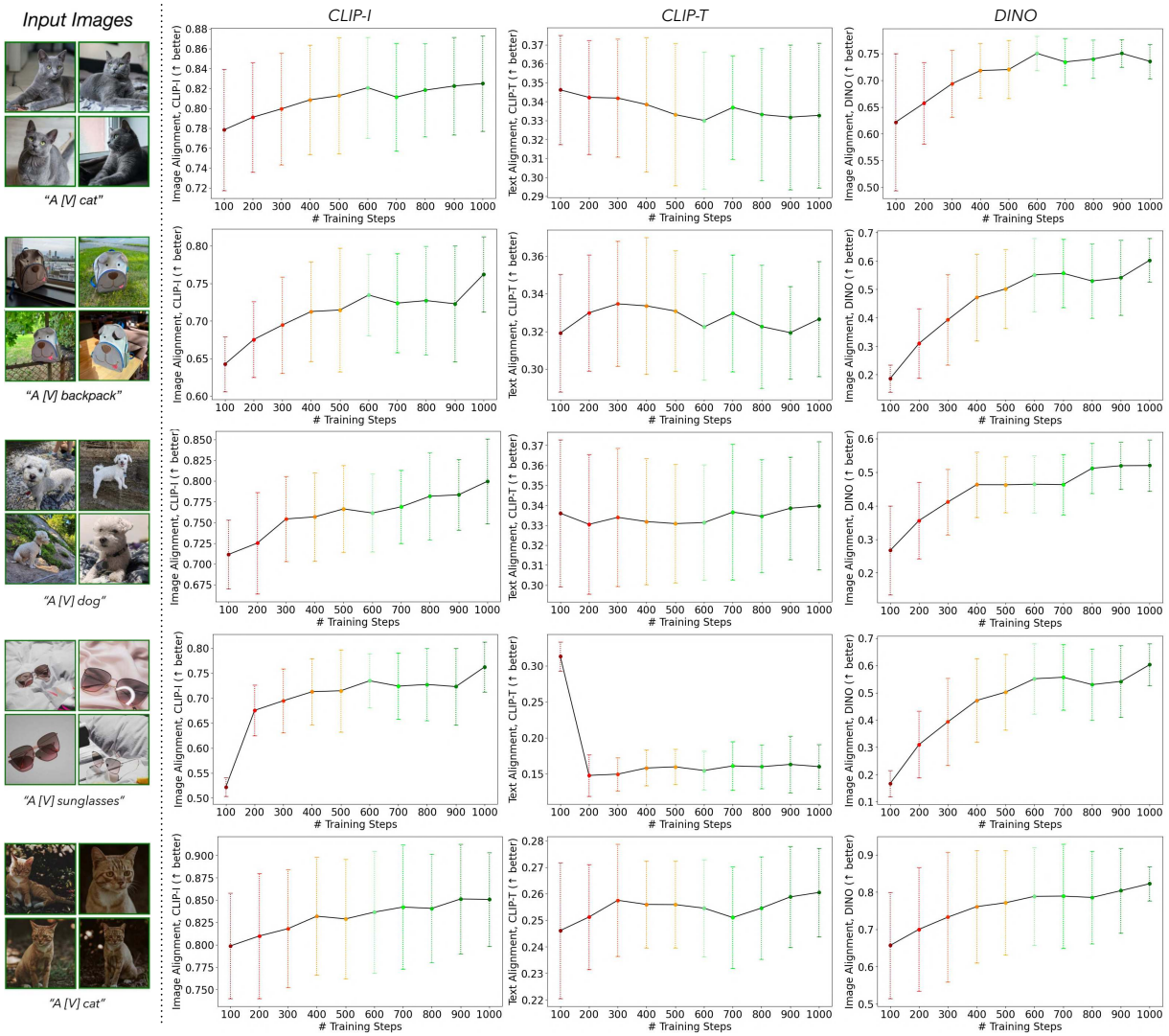


Figure 10. Plots depicting image alignment, text alignment, and DINO scores against training iterations. The scores are computed from the same set of images and prompts as depicted in Fig. 9.

very high. However, in cases where the input image complexity or the prompt complexity requires additional refinement, it is better to extend the training up to 1000 iterations as depicted in Fig. 9 and Fig. 10.

The images generated by *DiffuseKronA* show a clear progression in quality with respect to different steps. As the steps increase, the model seems to refine the details and improve the quality of the images. This iterative process allows the model to gradually improve the image, adding more details and making it more accurate to the prompt.

In Fig. 9 for instance, *"A cat floating on water in a swimming pool"*, in the initial iterations, the model generates a basic image of a cat floating on water. As the iterations progress and reach 500, the model refines the image, adding more details such as the color and texture of the cat, the ripples in the water, and the details of the swimming pool. At 1000 steps the image is a detailed and realistic representation of the prompt.

In Fig. 9, *"A backpack on top of a white rug"*, the early iterations produce a simple image of a backpack on a white surface. However, as the iterations increase, the model adds more details to the backpack, such as the zippers, pockets, and straps. It also starts to add texture to the white rug, making it look more realistic. By the final iteration, the white rug gets smoother in texture producing a fine image.

4.5. Effect of the number of training images

4.5.1 One shot image generation

The images are high-quality and accurately represent the text prompts. They are clear and well-drawn, and the content of each image matches the corresponding text prompt

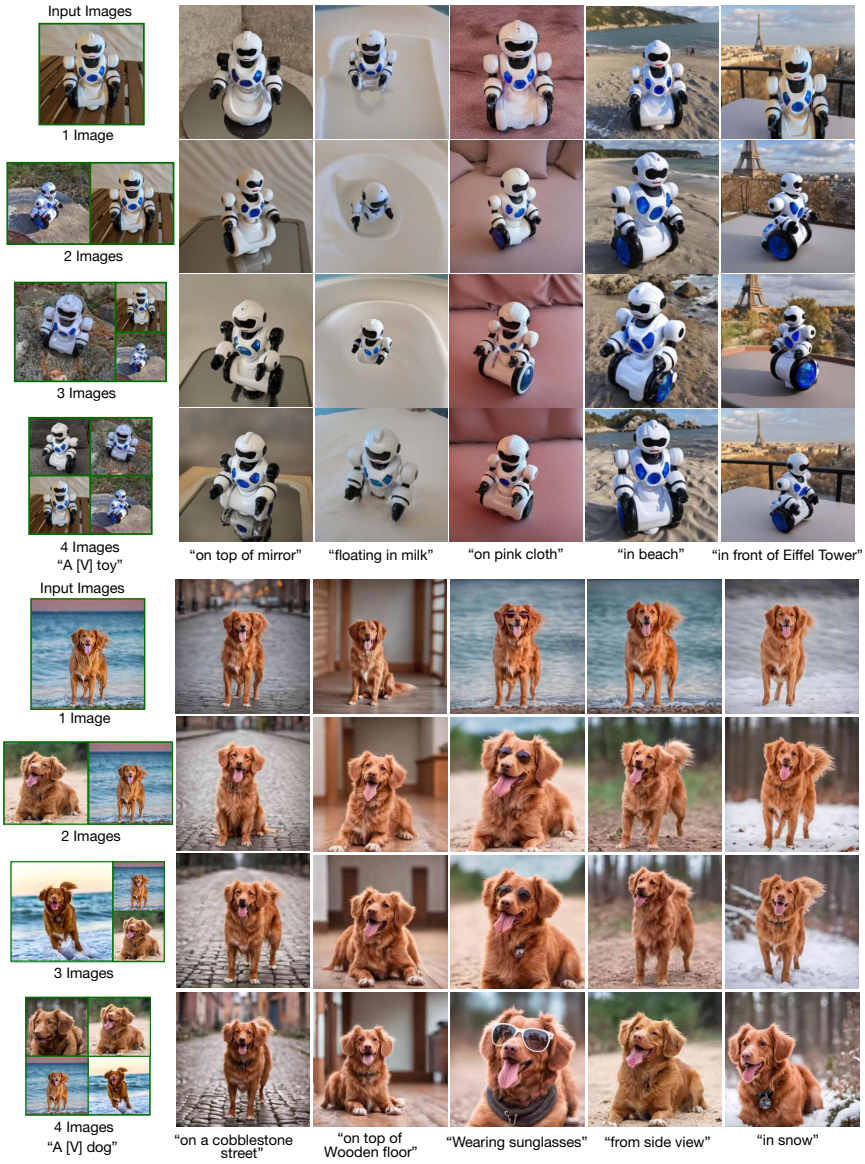


Figure 11. The influence of training images on fine-tuning. Even though *DiffuseKronA* produces impressive results with a single image, the generation of images with a broader range of perspectives is enhanced when more training images are provided with variations.

perfectly. For instance, in Fig. 4 (of main paper), the image of the *"A [V] logo"* is a yellow smiley face with hands. The *"made as a coin"* prompt resulted in a grey ghost with a white border, demonstrating the model's ability to incorporate abstract concepts. The *"futuristic neon glow"* and *"made with water colours"* prompts resulted in a pink and a yellow octopus respectively, showcasing the model's versatility in applying different artistic styles. The model's ability to generate an image of a guitar-playing octopus on a grey notebook from the prompt *"sticker on a notebook"* is a testament to its advanced capabilities.

The images are diverse in terms of style and content which is impressive, especially considering that these images were generated in a one-shot setting which makes it suitable for image editing tasks. While our model demonstrates remarkable proficiency in generating compelling results with a single input image, it encounters challenges when attempting to generate diverse poses or angles. However, when supplied with multiple images (2, 3, or 4), our model adeptly captures additional spatial features from the input images, facilitating the generation of images with a broader range of poses and angles. Our model can effectively use the information from multiple input images to generate more accurate and detailed output images as depicted in Fig. 11.

Figure 12. Images produced by adjusting the guidance score (α) reveal that a score of 7 produces the most realistic results. Increasing the score beyond 7 significantly amplifies the contrast of the images.

Figure 13. The influence of inference steps on image generation. Optimal results are achieved in the range of 50-70 steps, striking a balance between textual input and subject fidelity. Here, we opted for 50 inference steps to minimize inference time.

4.6. Effect of Inference Hyperparameters

Guidance Score (α). The guidance score, denoted as α, regulates the variation and distribution of colors in the generated images. A lower guidance score produces a more subdued version of colors in the images, aligning with the description provided in the input text prompt. In contrast, a higher guidance score results in images with more vibrant and pronounced colors. Guidance scores ranging from 7 to 10 generally yield images with an appropriate and well-distributed color palette.

In the example of *"A [V] toy"* in Fig. 12, when the prompt is *"made of purple color"*, it is evident that a reddish lavender hue is generated for a guidance score of 1 or 3. Conversely, with a guidance score exceeding 15, a mulberry shade is produced. For guidance scores close to 8, images with a pure purple color are formed.

Number of inference Steps. The number of steps plays a crucial role in defining the granularity of the generated images. As illustrated in Fig. 13, during the initial steps, the model creates a subject that aligns with the text prompt and begins incorporating features from the input image. With the progression of generation, finer details emerge in the images. Optimal results, depending on the complexity of prompts, are observed within the range of 30 to 70 steps, with an average of 50 steps proving to be the most effective. However, exceeding 100 steps results in the introduction of noise and a decline in the quality of the generated images.

The quality of the generated images appears to improve with an increase in the number of inference steps. For instance, the images for the prompt *"a toy"* and *"wearing sunglasses"* appear to be of higher quality at 50 and 75 inference steps respectively, compared to at 10 inference steps.

5. Detailed study on LoRA-DreamBooth vs *DiffuseKronA*

In this section, we expand our analysis of model performance by comparing LoRA-DreamBooth and *DiffuseKronA* across various aspects, including fidelity, color distribution, text alignment, stability, and complexity.

5.1. Fidelity & Color Distribution

DiffuseKronA generates images of superior fidelity as compared to LoRA-DreamBooth in lieu of the higher representational power of Kronecker Products along with its ability to capture spatial features.

In the example of *"A [V] backpack"* in Fig. 14, the following observations can be made:

(1) ***"with the Eiffel Tower in the background"***: The backpack generated by *DiffuseKronA* is pictured with the Eiffel Tower in the background, creating a striking contrast between the red of the backpack and the muted colors of the cityscape, which LoRA-DreamBooth fails to do.

(2) ***"city in background"***: The backpack generated by *DiffuseKronA* is set against a city backdrop, where the red color stands out against the neutral tones of the buildings, whereas, LoRA-DreamBooth does not generate high contrast between images.

(3) ***"on the beach"***: The image generated by *DiffuseKronA* shows the backpack on a beach, where the red contrasts with the blue of the water and the beige of the sand.

5.2. Text Alignment

DiffuseKronA is more accurate in aligning text with images compared to the Lora-DreamBooth. For instance, in the first row, *DiffuseKronA* correctly aligns the text with *"sunflowers inside"* with the image of a vase with sunflowers, whereas LoRA-DreamBooth fails to align the sunflower in the vase of the same color as of input images.

In more complex input examples like in Fig. 15, such as the one involving anime in *"A [V] character"*, the generated images by LoRA-DreamBooth lack the sense of cooking a meal and a karaoke bar, whereas *DiffuseKronA* consistently produces images that closely align with the provided text prompts.

5.3. Complex Input images and Prompts

DiffuseKronA demonstrates a notable emphasis on capturing nuances within text prompts and excels in preserving intricate details from input images to the highest degree. In contrast, LoRA-DreamBooth lacks these properties. This distinction is evident in Fig. 16, where, for the prompt *"A [V] face"*, *DiffuseKronA* successfully generates an ivory-white blazer and a smiling face, while LoRA-DreamBooth struggles to maintain both the color and the smile on the face.

Similarly, for the prompt *"A [V] clock"* in Fig. 16, *DiffuseKronA* accurately reproduces detailed numbers, particularly 3, from the input images. Although it encounters challenges in preserving the structure of numbers while creating a clock of cubical shape, it still maintains a strong focus on text details— a characteristic lacking in LoRA-DreamBooth.

5.4. Qualitative and Quantitative comparison

We have assessed the image generation capabilities of *DiffuseKronA* and LoRA-DreamBooth on SDXL [19]. Our findings reveal that *DiffuseKronA* excels in generating images with high fidelity, more accurate color distribution, and greater stability compared to LoRA-DreamBooth.

6. Comparison with other Low-Rank Decomposition methods

In this section, we compare our *DiffuseKronA* with low-rank methods other than LoRA, specifically with LoKr [27]

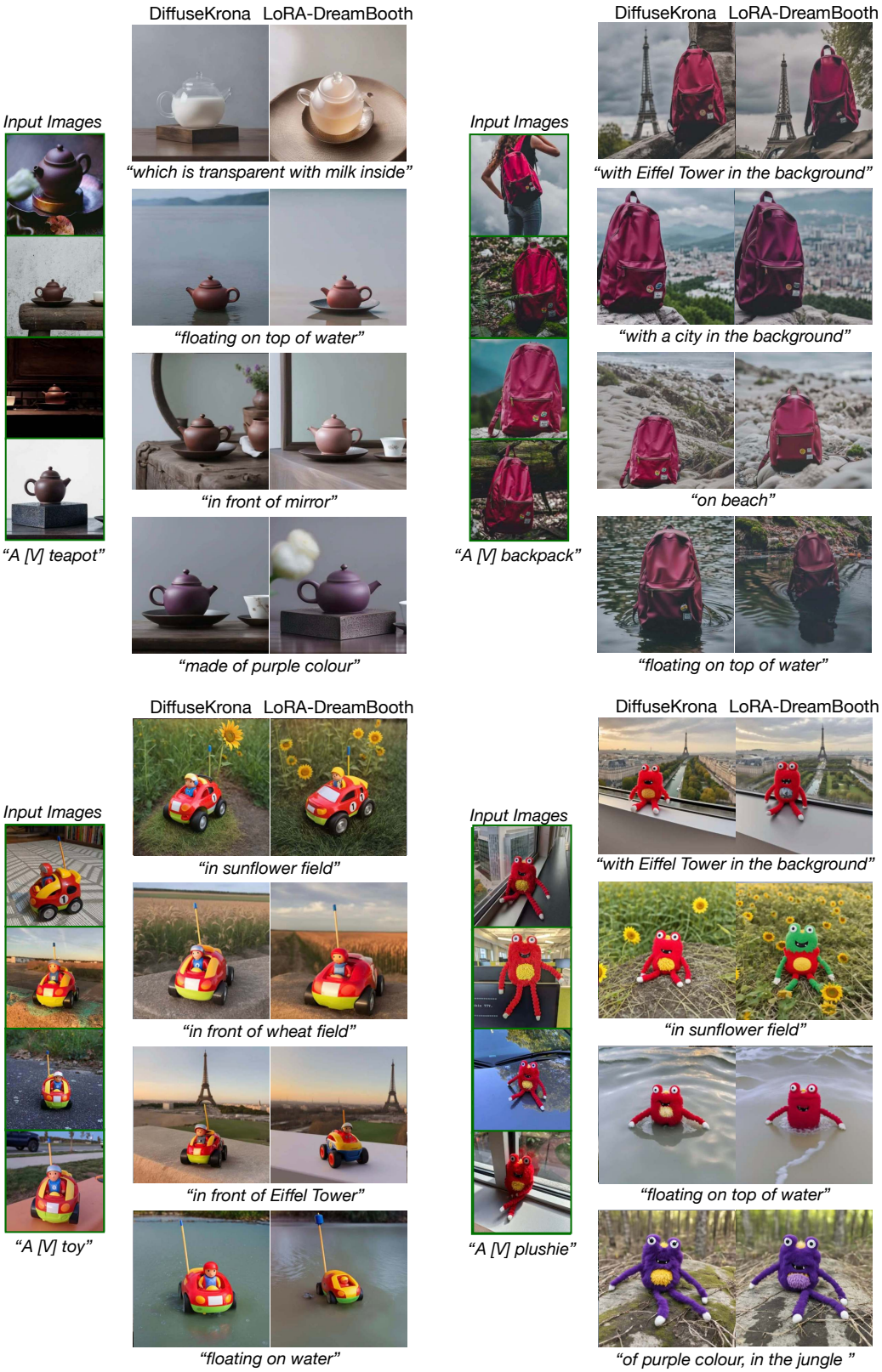


Figure 14. Comparison of fidelity and color preservation in *DiffuseKronA* and LoRA-DreamBooth.

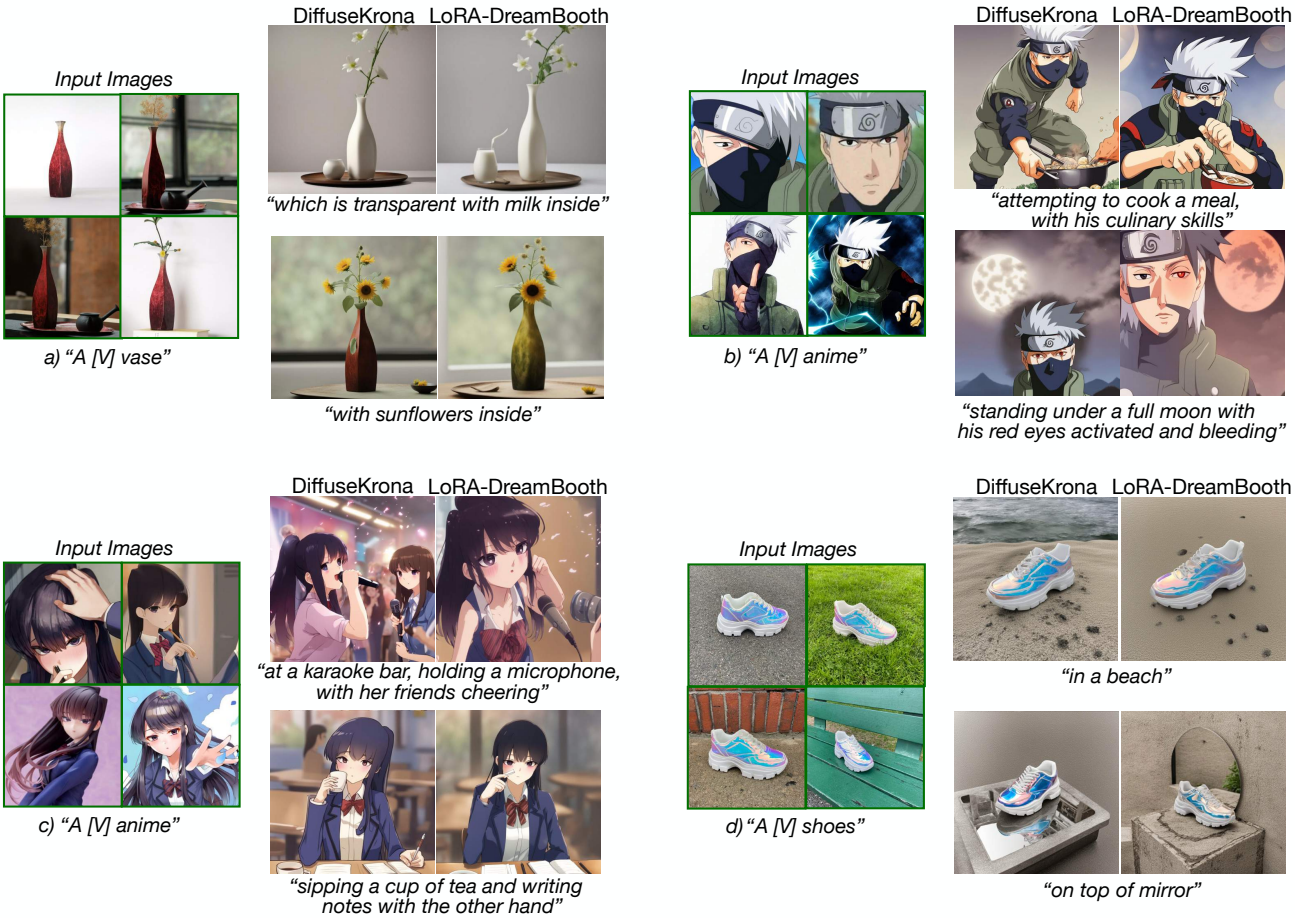


Figure 15. Comparison of text alignment in generated images by our proposed *DiffuseKronA* and LoRA-DreamBooth.

and LoHA [27]. We also note that our implementation is independent of the `LyCORIS` project [27], and we did not use LoKr nor LoHA in *DiffuseKronA*[1]. We summarize the key differences between *DiffuseKronA* and these methods as follows:

❶ *DiffuseKronA* has 2 controllable parameters (a_1 and a_2), which are chosen manually through extensive experiments (refer to Fig. 7 and Tab. 2), whereas LoKr [27] follows the procedure mentioned in the FACTORIZATION function (see right) which depends on input dimension and another hyper-parameter called *factor*. Following the descriptions on the implementation of Figure 2 in [27], and we quote "we set the factor to 8 and do not perform further decomposition of the second block", the default implementation makes A a square matrix of dimension $(factor \times factor)$. Notably, for any factor, $f > 0$, A would always be a square matrix of shape $(f \times f)$ which is a special case (a subset) of *DiffusKronA* (diagonal entry in Fig. 7) but for $f = -1$, A matrix size would be completely dependent upon dimension, and it would not be a square matrix always.

[1]To ensure a fair comparison, we have incorporated LoKr and LoHA into the SDXL backbone.

```python
def factorization(dim: int, factor: int=-1):
    if factor > 0 and (dim % factor) == 0:
        m = factor
        n = dim // factor
        if m > n:
            n, m = m, n
        return m, n
    if factor < 0:
        factor = dim
    m, n = 1, dim
    length = m + n
    while m < n:
        new_m = m + 1
        while dim % new_m != 0:
            new_m += 1
        new_n = dim // new_m
        if new_m + new_n > length or new_m > factor:
            break
        else:
            m, n = new_m, new_n
    if m > n:
        n, m = m, n
    return m, n
```

Listing 1. This code snippet is extracted from the official `LyCORIS` codebase (Link).

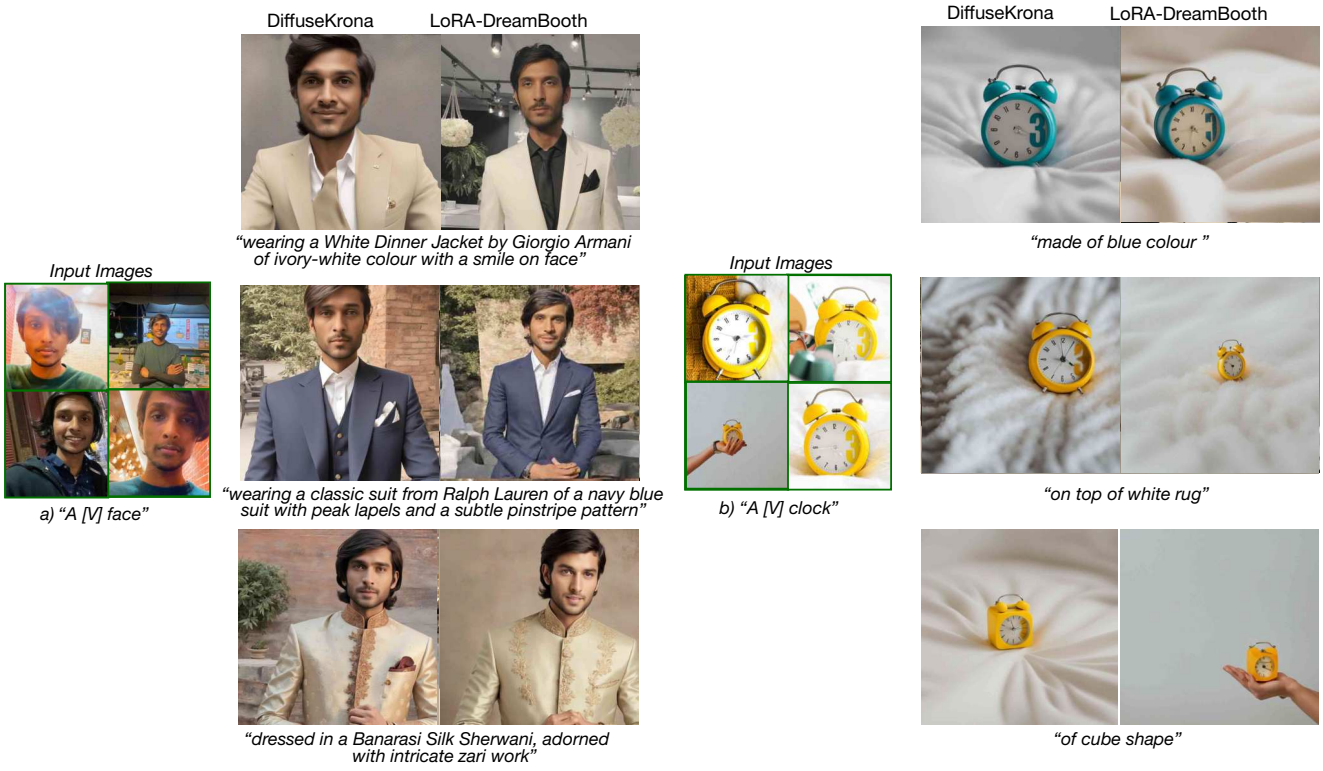


Figure 16. Comparison of image generation on complex prompts and input images by *DiffuseKronA* and LoRA-DreamBooth.

These attributes make our way of performing Kronecker decomposition a superset of LoKr, offering greater control and flexibility compared to LoKr. On the other hand, LoHA has only one controllable parameter, *i.e.*, rank, similar to LoRA.

❷ LoKr takes the generic form of $\Delta W = A \otimes (B \cdot C)$, and LoHA adopts $\Delta W = (A \cdot B) \odot (C \cdot D)$, where $\odot$ denotes the Hadamard product. For more details, we refer the readers to Figure 1 in [27]. Based on the definition, LoHA does not explore the benefits of using Kronecker decomposition.

❸ [27] provided the first use of Kronecker decomposition in Diffusion model fine-tuning but limited analysis in the few-shot T2I personalization setting. In our study, we conducted detailed analysis and exploration to demonstrate the benefits of using Kronecker decomposition. Our new insights include large-scale analysis of parameter efficiency, enhanced stability to hyperparameters, and improved text alignment and fidelity, among others.

❹ We further compare our *DiffuseKronA* with LoKr and LoHA using the default implementations from [27] in Fig. 17 and Fig. 18, respectively. However, the default settings were used in the SD variant, and it is also evident that personalized T2I generations are very sensitive to model settings and hyper-parameter choices. Bearing these facts, we also explored the hyperparameters in both adapters. In Fig. 19, we have presented the ablation study examining the factors and ranks for LoKr utilizing SDXL, while in Fig. 20, we showcase an ablation study on the learning rate. Moreover, Fig. 21 features an ablation study on the learning rate and rank for LoHA using SDXL. These analyses reveal that for LoKr, the optimal factor is -1 and the optimal rank is 8, with a learning rate of 1×10^{-3}; while for LoHA, the optimal rank is 4, with a learning rate of 1×10^{-4}.

Additionally, quantitative comparisons are conducted, encompassing parameter count alongside image-to-image and image-to-text alignment scores, as detailed in Tab. 3 and Tab. 4. The results in Tab. 3 indicate that although LoKr marginally possesses fewer parameters still *DiffuseKronA* with $a_1 = 16$ achieves superior CLIP-I, CLIP-T, and DINO scores. This contrast is readily noticeable in the visual examples depicted in Fig. 17. For the prompt *"A [V] toy with the Eiffel Tower in the background"*, LoKr fails to construct the *Eiffel Tower* in the background, unlike *DiffuseKronA* ($a_1 = 16$). Similarly, in the case of *"A [V] teapot floating on top of water"* LoKr distorts the teapot's spout, whereas *DiffuseKronA* maintains fidelity. In the case of *"A [V] toy"* (last row), the results of *DiffuseKronA* are much more aligned as compared to LoKr for both prompts. Conversely, for *dog* and *cat* examples, all the methods demonstrate similar visual appearance in terms of fidelity as well as textual alignment. Consequently, it's evident that while LoKr reduces parameter count, it struggles

with complex input images or text prompts with multiple contexts. Hence, *DiffusekronA* achieves efficiency in parameters while upholding average scores across CLIP-I, CLIP-T, and DINO metrics. Hence, achieving a better trade-off between parameter efficiency and personalized image generation.

Table 3. **Quantitative comparison** of *DiffuseKronA* with low-rank decomposition methods namely LoRA, LoKr, and LoHA in terms of the number of trainable parameters, text-alignment, and image-alignment scores. The scores are computed from the same set of images and prompts as depicted in Fig. 17.

MODEL	# PARAMETERS (↓)	CLIP-I (↑)	CLIP-T (↑)	DINO (↑)
DiffuseKronA $a_1 = 2$	3.8 M	0.799 ±0.073	0.267 ±0.048	0.648 ±0.122
DiffuseKronA $a_1 = 4$	7.5 M	0.809 ±0.086	0.268 ±0.055	0.651 ±0.142
DiffuseKronA $a_1 = 8$	2.1 M	0.815 ±0.074	0.313 ±0.024	0.649 ±0.139
DiffuseKronA $a_1 = 16$	1.5 M	0.817 ±0.078	0.301 ±0.038	0.654 ±0.127
LoRA-DreamBooth $rank = 4$	5.8 M	0.807 ±0.077	0.288 ±0.033	0.635 ±0.136
LoKr $f = -1, rank = 8$	**1.36 M**	0.801 ±0.065	0.287 ±0.049	0.646 ±0.147
LoKr $f = 8$	14.9 M	0.812 ± 0.069	0.277 ±0.042	0.639 ±0.111
LoHA $rank = 4$	20.9 M	0.818 ±0.064	0.299 ±0.041	0.641 ±0.120

Table 4. **Quantitative comparison** of *DiffuseKronA* with varying factors (*i.e.* 2, 4, 8, 16) of LoKr in terms of the number of trainable parameters, text-alignment, and image-alignment scores. The scores are computed from the same set of images and prompts as depicted in Fig. 18.

MODEL	# PARAMETERS (↓)	CLIP-I (↑)	CLIP-T (↑)	DINO (↑)
LoKr $f = 2$	238.7 M	0.825 ±0.037	0.244 ±0.024	0.727 ±0.036
LoKr $f = 4$	59.7 M	0.784 ± 0.063	0.246 ±0.030	0.683 ±0.051
LoKr $f = 8$	14.9 M	0.749 ±0.067	0.292 ±0.064	0.568 ±0.075
LoKr $f = 16$	3.8 M	0.707 ± 0.121	0.231 ±0.025	0.472 ±0.160
DiffuseKronA $a_1 = 8$	**2.1 M**	0.806 ± 0.028	0.281 ± 0.070	0.653 ± 0.045

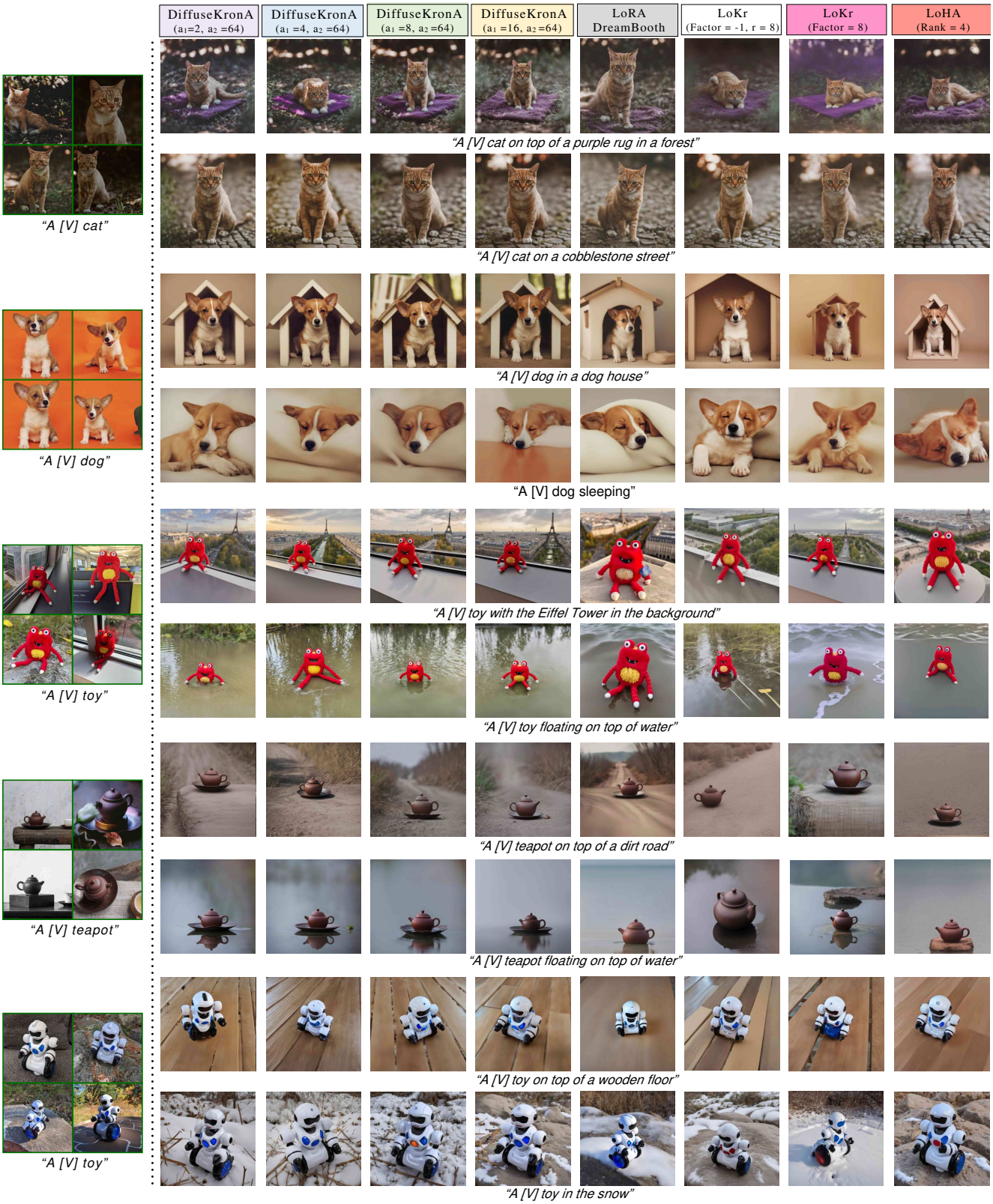


Figure 17. **Qualitative comparison** of four variants of *DiffusekronA* with other low-rank methods including LoRA, LoKr, and LoHA. Learning rates: *DiffusekronA* (5×10^{-4}), LoRA (1×10^{-4}), LoKr (1×10^{-3}) & LoHA (1×10^{-4}).

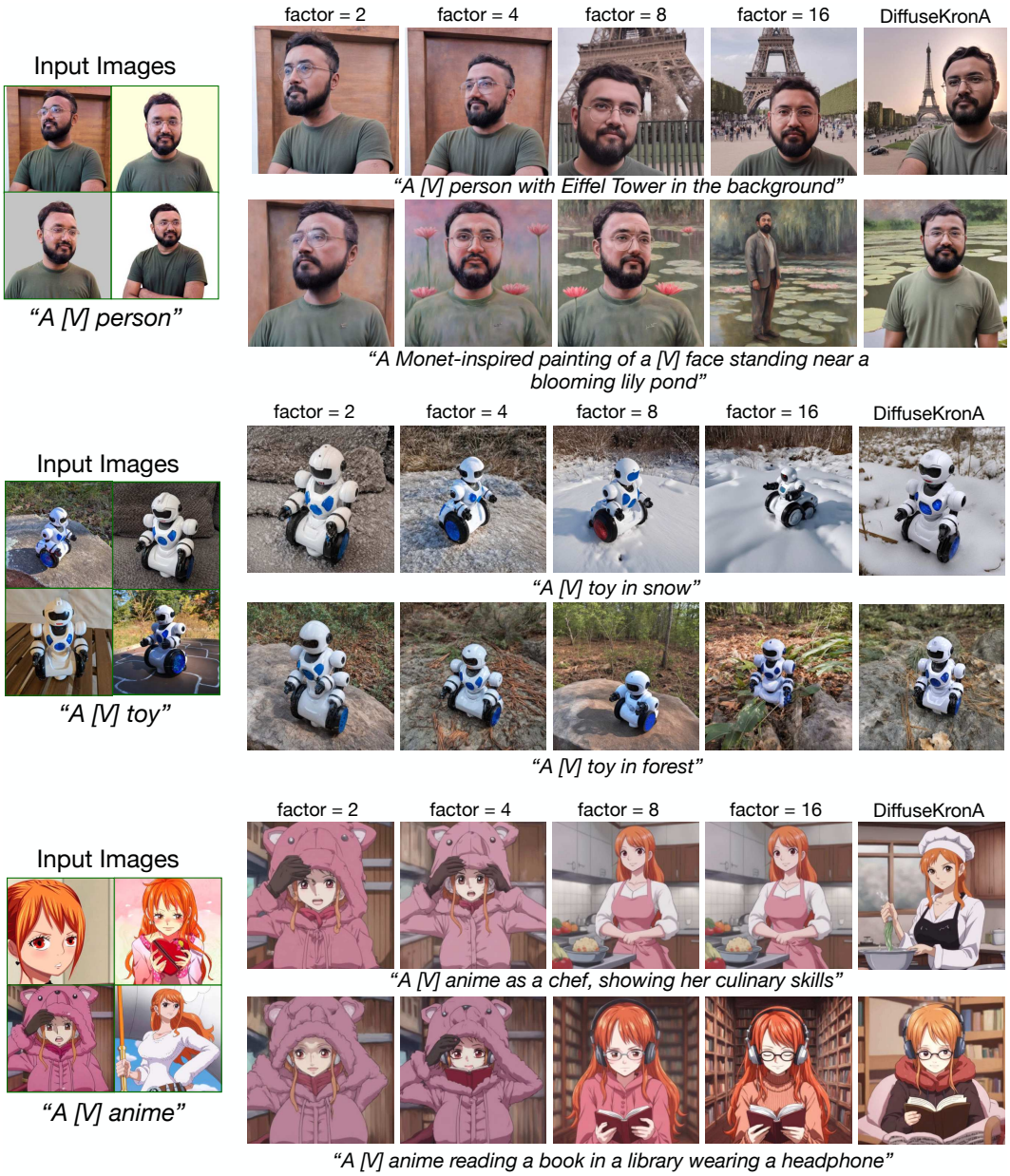


Figure 18. **Qualitative comparison.** Results are shown for the default factors given by the LoKr implementation, with the varying factors being 2, 4, 8, and 16.

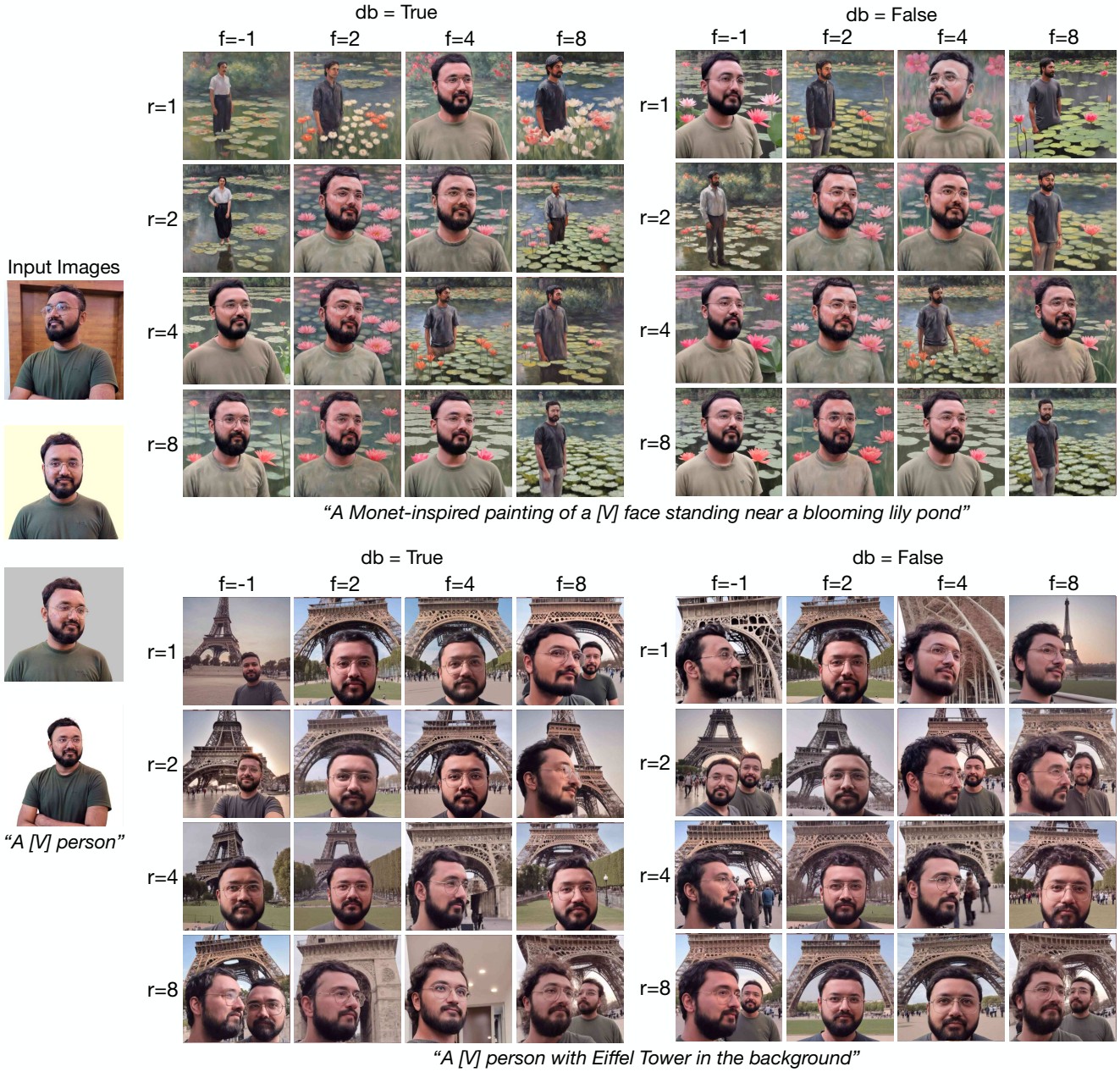


Figure 19. **Ablation study on factor and rank** for LoKr using SDXL, with a learning rate of 1×10^{-3}. We found that the optimal factor and rank are -1 and 8, respectively. We also experimented with db=True, which indicates further low-rank decomposition of both matrices A and B, whereas db=False means only matrix B is decomposed further. (**continued**..)

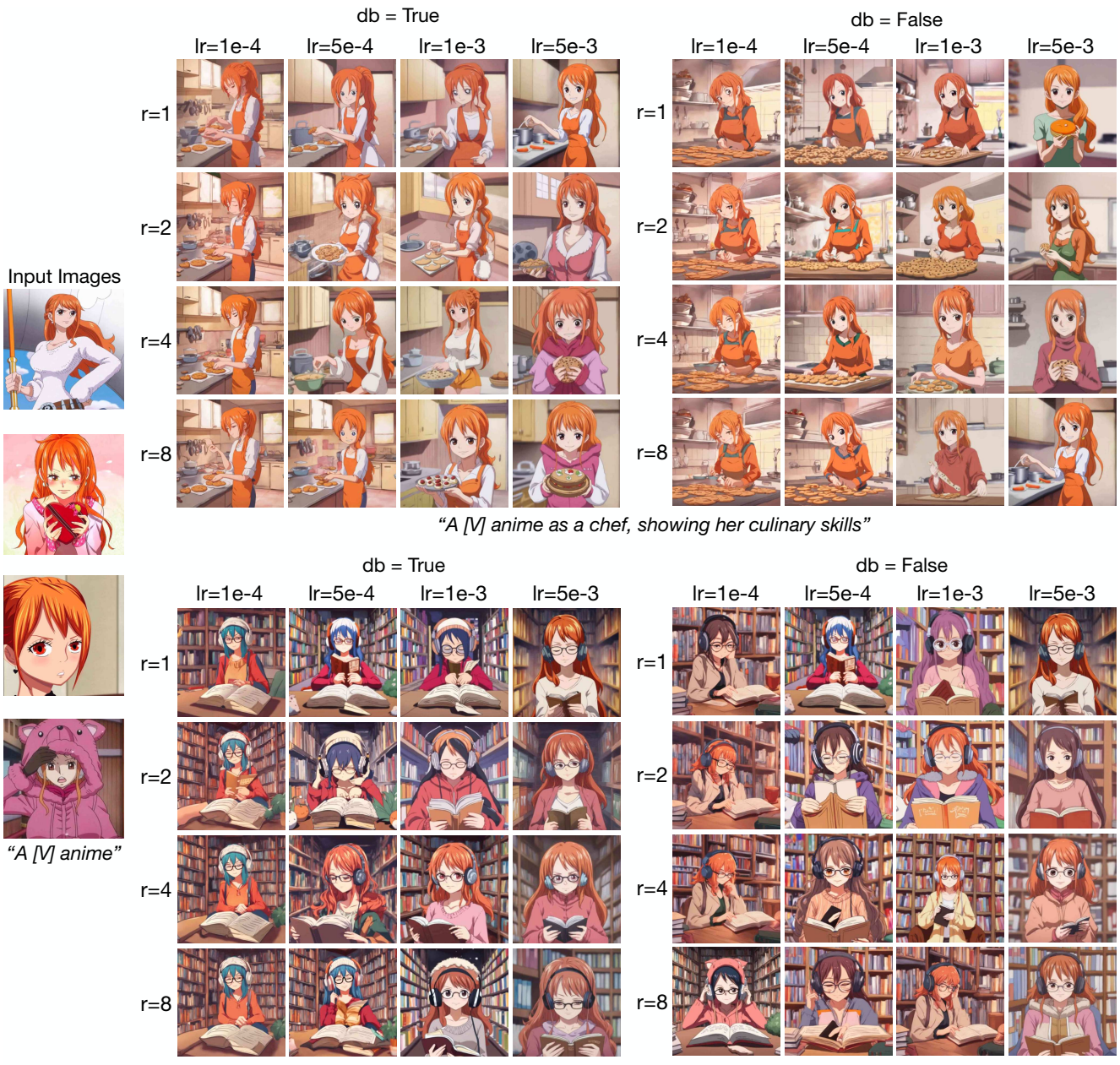


Figure 19. **Ablation study on factor and rank** for LoKr using SDXL, with a learning rate of 1×10^{-3}. We found that the optimal factor and rank are -1 and 8, respectively. We also experimented with db=True, which indicates further low-rank decomposition of both matrices A and B, whereas db=False means only matrix B is decomposed further. (**continued**..)

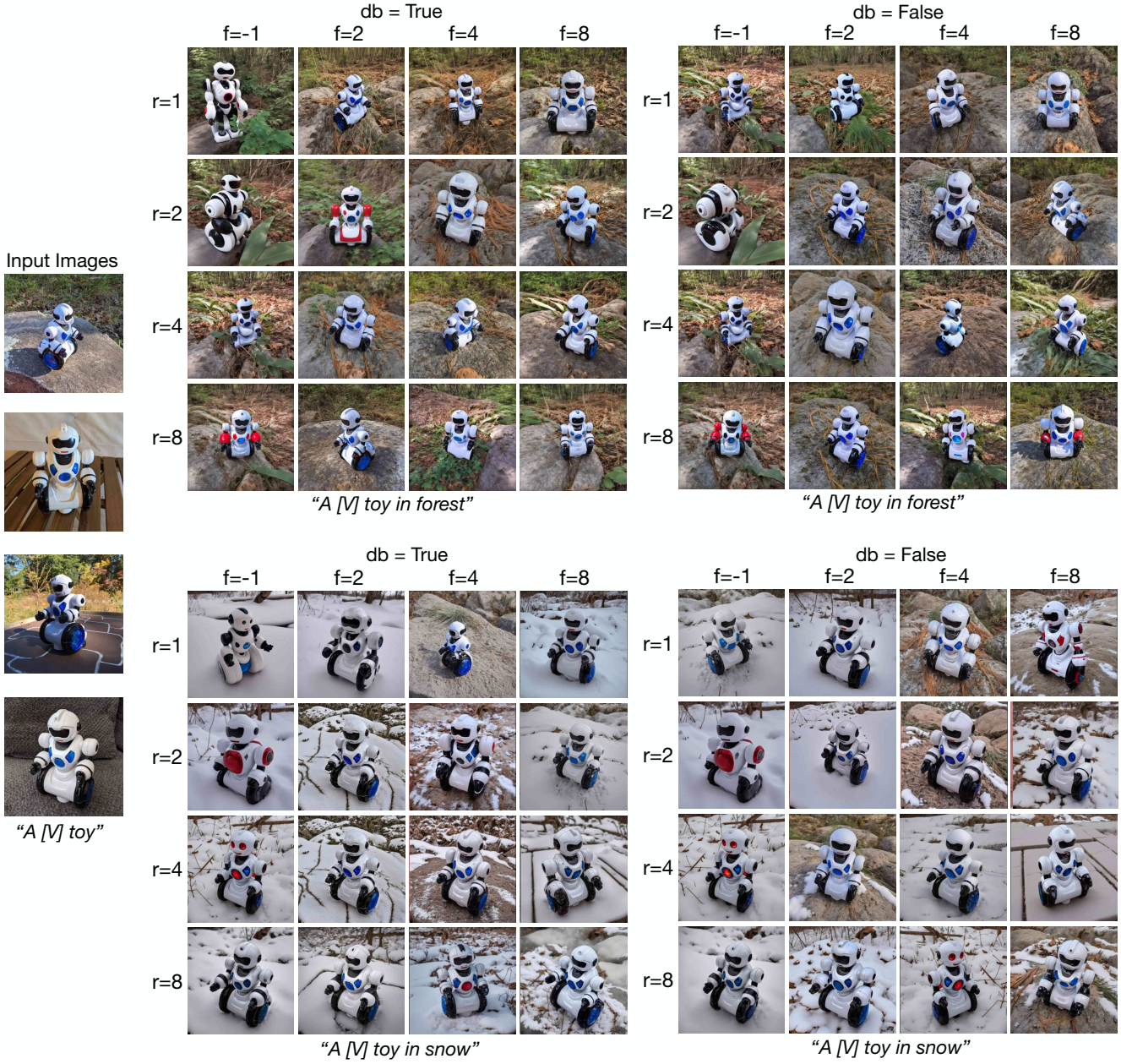


Figure 19. **Ablation study on factor and rank** for LoKr using SDXL, with a learning rate of 1×10^{-3}. We found that the optimal factor and rank are -1 and 8, respectively. We also experimented with db=True, which indicates further low-rank decomposition of both matrices A and B, whereas db=False means only matrix B is decomposed further. (**end**)

Figure 20. **Ablation study on factor and learning rate** for LoKr using SDXL, with a fixed factor of -1. We found that the optimal learning rate and rank are 1×10^{-3} and 8, respectively. We also experimented with db=True, which indicates further low-rank decomposition of both matrices A and B, whereas db=False means only matrix B is decomposed further. (**continued**..)

Figure 20. **Ablation study on factor and learning rate** for LoKr using SDXL, with a fixed factor of -1. We found that the optimal learning rate and rank are 1×10^{-3} and 8, respectively. We also experimented with db=True, which indicates further low-rank decomposition of both matrices A and B, whereas db=False means only matrix B is decomposed further. (**continued**..)

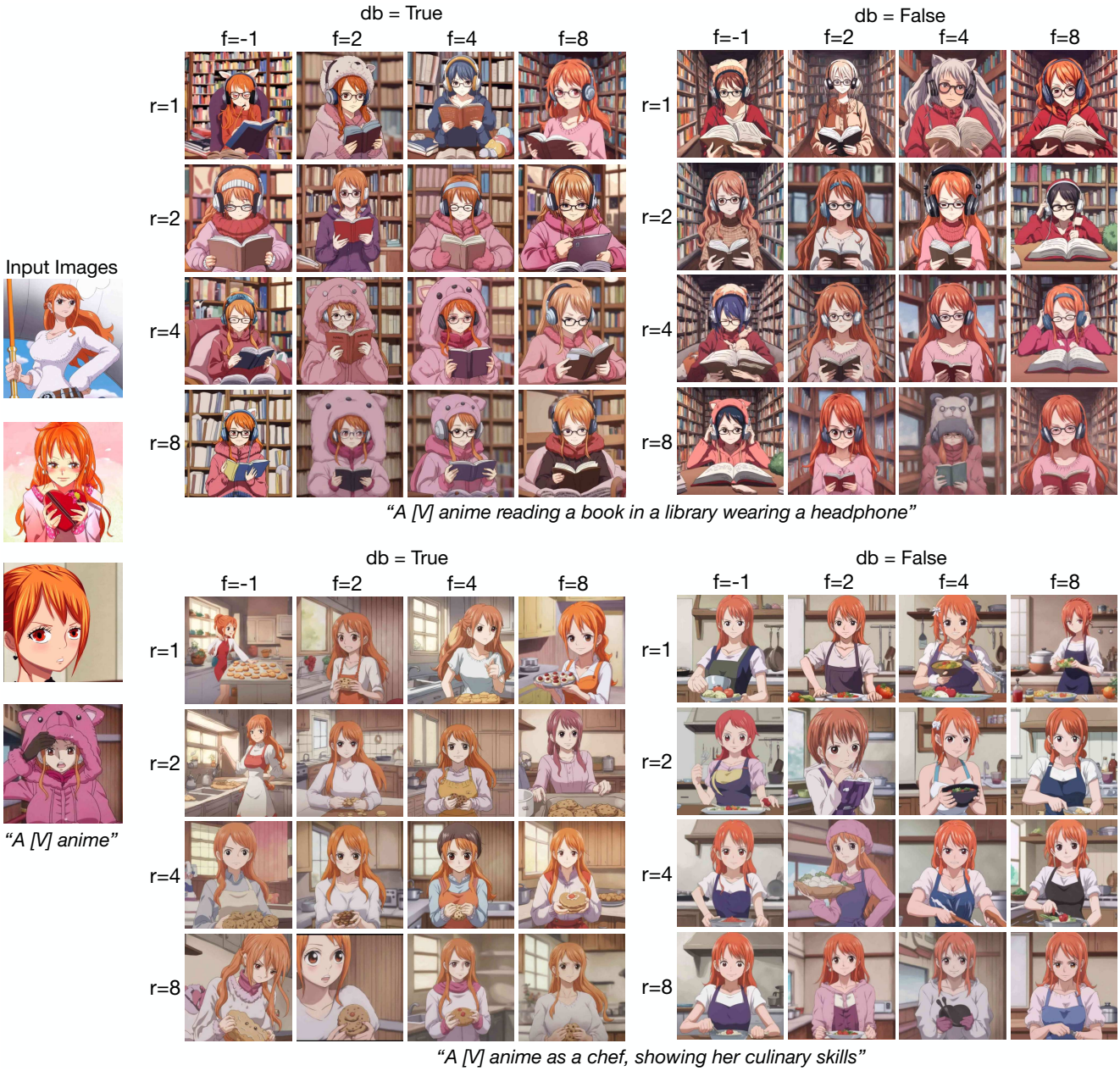


Figure 20. **Ablation study on factor and learning rate** for LoKr using SDXL, with a fixed factor of -1. We found that the optimal learning rate and rank are 1×10^{-3} and 8, respectively. We also experimented with db=True, which indicates further low-rank decomposition of both matrices A and B, whereas db=False means only matrix B is decomposed further. (**end**)

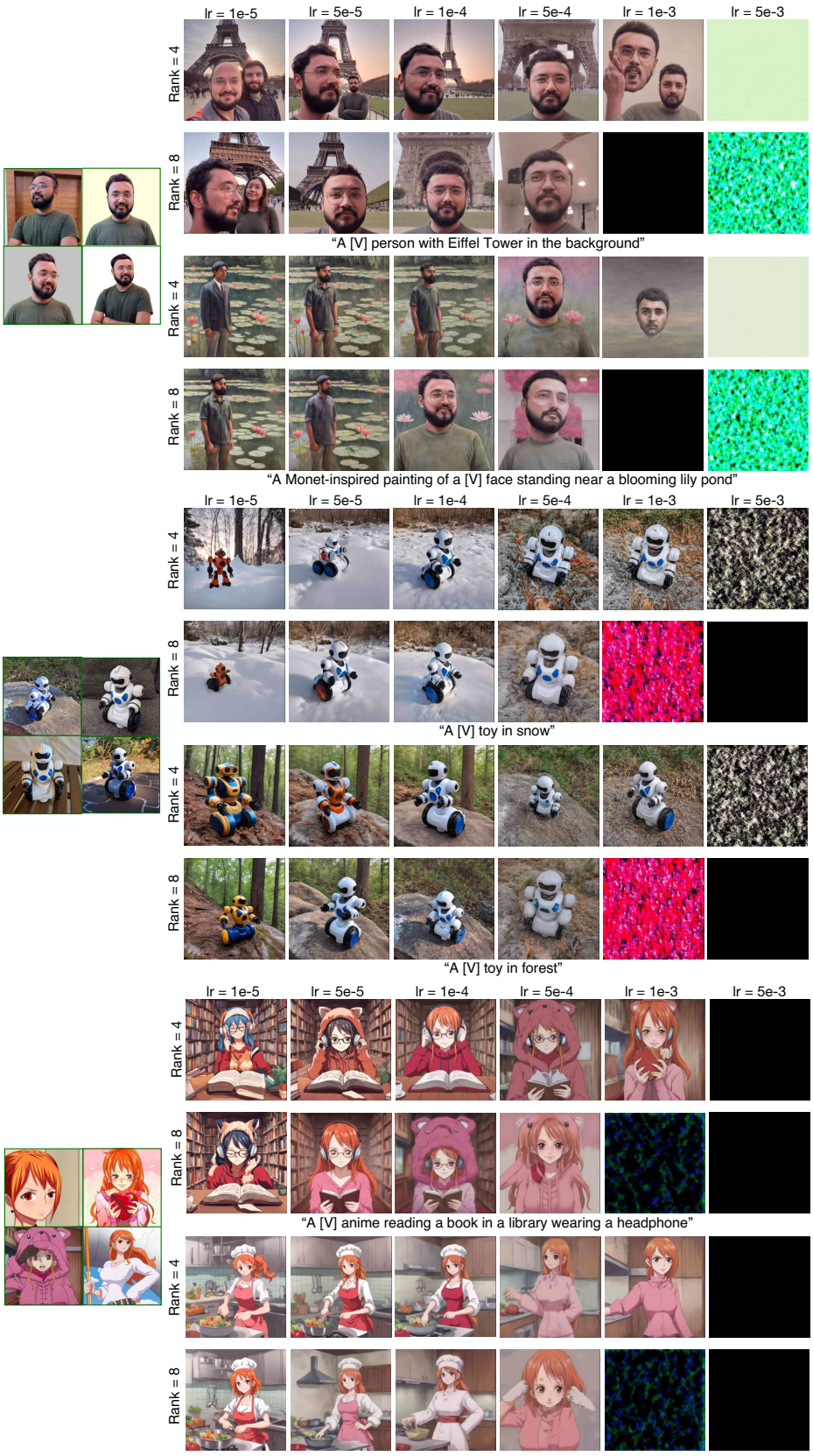


Figure 21. Ablation study on learning rate and rank for LoHA using SDXL. We found the optimal learning rate and rank to be 1×10^{-4} and 4, respectively.

7. Comparison with state-of-the-arts

Qualitative Comparison. In this section, we extend upon Sec. 4.4. of the main paper, comparing *DiffuseKronA* with State-of-the-art text-to-image personalization models including DreamBooth, LoRA-DreamBooth, SVDiff, Textual Invention, and Custom Diffusion.

(**1**) Textual Inversion [7] is a fine-tuning method that optimizes a placeholder embedding to reconstruct the training set of subject images. Learning a new concept requires 3,000 steps, which takes around 30 minutes on an A100 GPU [15].

(**2**) DreamBooth [22] refines the entire network through additional preservation loss as a form of regularization, leading to enhancements in visual quality that exhibit promising results. Updating DreamBooth for a new concept typically requires about 6 minutes on an A100 GPU [15].

(**3**) LoRA-DreamBooth [23] explores low-rank adaptation for parameter-efficient fine-tuning attention-weight matrices of the text-to-image diffusion model. Fine-tuning LoRA-DreamBooth for a new concept typically takes about 5 minutes on a single 24GB NVIDIA RTX-3090 GPU.

(**4**) SVDiff [9] involves fine-tuning the singular values of the weight matrices, leading to a compact and efficient parameter space that reduces the risk of overfitting and language drifting. It took around 15 minutes on a single 24GB NVIDIA RTX-3090 GPU[2].

(**5**) Custom diffusion [13] involves selective fine-tuning of weight matrices through a conditioning mechanism, enabling parameter-efficient refinement of diffusion models. This approach is further extended to encompass multi-concept fine-tuning. The fine-tuning time of Custom diffusion is around 6 minutes on 2 A100 GPUs.

Qualitative Comparison. *DiffuseKronA* consistently produces images closely aligned with the input images and consistently integrates features specified in the input text prompt. The enhanced fidelity and comprehensive comprehension of the input text prompts can be attributed to the structure-preserving capability and improved expressiveness facilitated by Kronecker product-based adaptation. The images generated by LoRA-DreamBooth are not of high quality and demand extensive experimentation for improvement, as depicted in Fig. 22. As depicted in the figure, *DiffuseKronA* not only generates well-defined images but also has a better color distribution as compared to Custom Diffusion.

8. Comparison with tuning-free methods

Recent advancements within tuning-free methods obtain reasonable performance, some of them are even comparable with tuning-based methods in some aspects and are much more efficient. Here, we have evaluated the proposed method using all the samples and prompts from [22], so that some tuning-free methods [1, 15, 16, 18, 29] can also be compared under the same setting using results from corresponding papers, because some of them may not have official implementations. We have not included this in the main paper due to having different objectives but added it here to give more information to readers by providing a fair and comprehensive comparison. In Tab. 5, we quantitatively evaluate our method with recently proposed tuning and tuning-free methods.

Table 5. **Quantitative comparison** of subject fidelity (DINO and CLIP-I), prompt fidelity (CLIP-T) of *DiffuseKronA* with SOTA. The evaluation images and prompts are the same taken from `DreamBench` [22] (25 subjects with 30 text prompts for each subject).

	Method	CLIP-I (↑)	CLIP-T (↑)	DINO (↑)
Tuning-Free	SuTI [1]	0.819	0.304	0.741
	BLIP-Diffusion [15]	0.805	0.302	0.670
	Kosmos-G [18]	0.847	0.287	0.694
	CAFE [29]	0.827	0.294	0.715
	Subject-Diffusion [16]	0.787	0.293	0.711
Tuning	OFT [20]	0.785	0.237	0.632
	DreamBooth [22]	0.803	0.305	0.668
	DiffuseKronA (Ours)	0.814	0.306	0.712

9. Practical Implications

- Content Creation: It can be used to generate photorealistic content from text prompts.
- Image Editing and In-painting: The model can be used to edit images or fill in missing parts of an image.
- Super-Resolution: It can be used to enhance the resolution of images.
- Video Synthesis: The model can be used to generate videos from text prompts.
- 3D Assets Production: It can be used to create 3D assets from text prompts.
- Personalized Generation: The model can be used in personalized generation with DreamBooth fine-tuning.
- Resource Efficiency: The model is resource-efficient and can be trained with limited resources.
- Model Compression: The model allows for architectural compression, reducing the number of parameters, MACs per sampling step, and latency.

[2]SVDiff did not release official codebase, we used open-source code for SVDiff results in Fig. 22.

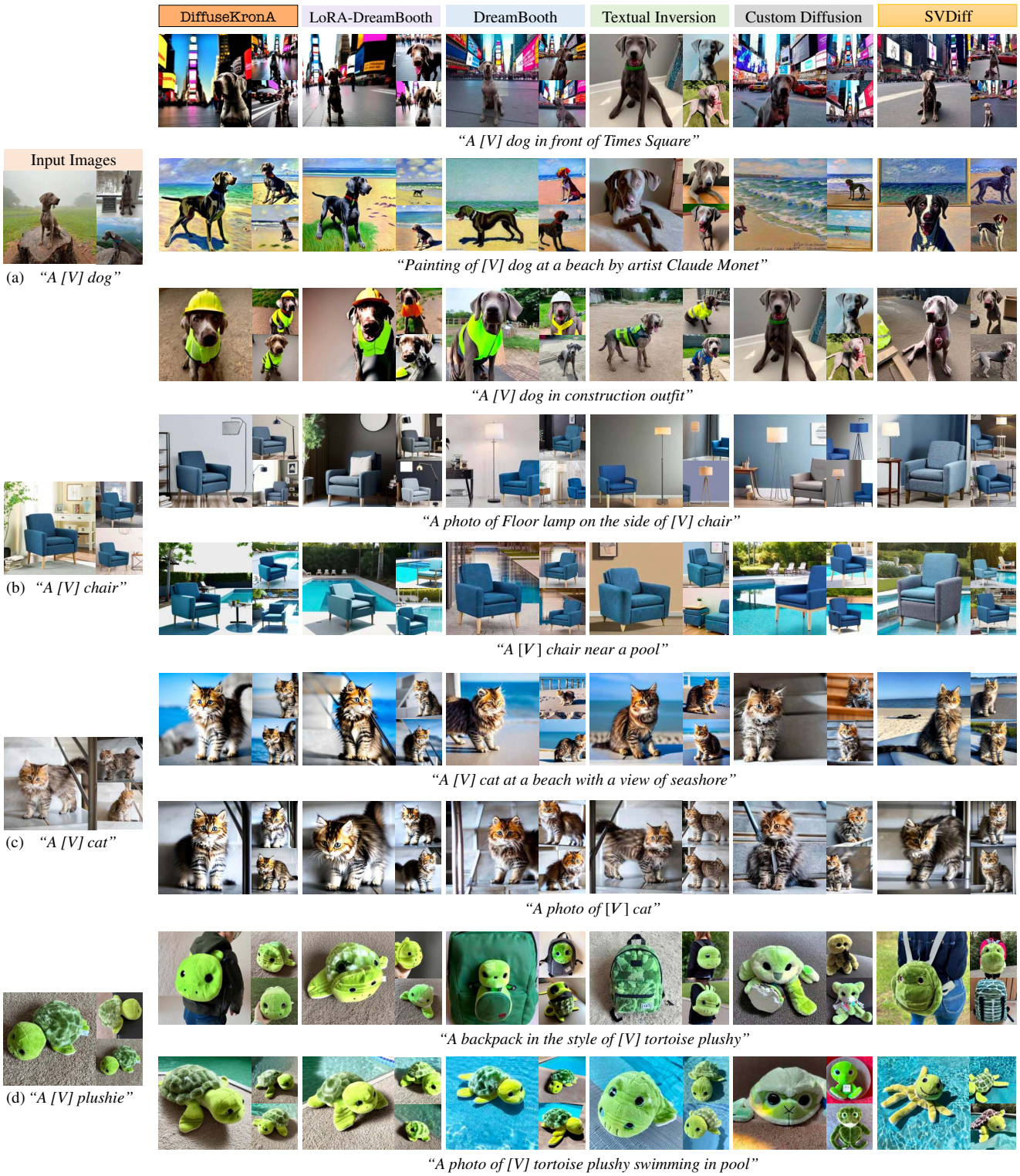


Figure 22. **Qualitative comparison** between generated images by *DiffuseKronA*, LoRA-DreamBooth, Textual Inversion, DreamBooth, and Custom Diffusion. Notably, our methods' results are generated considering $a_2 = 8$. We maintained the original settings of all these methods and used the SD CompVis-1.4 [2] variant to ensure a fair comparison.

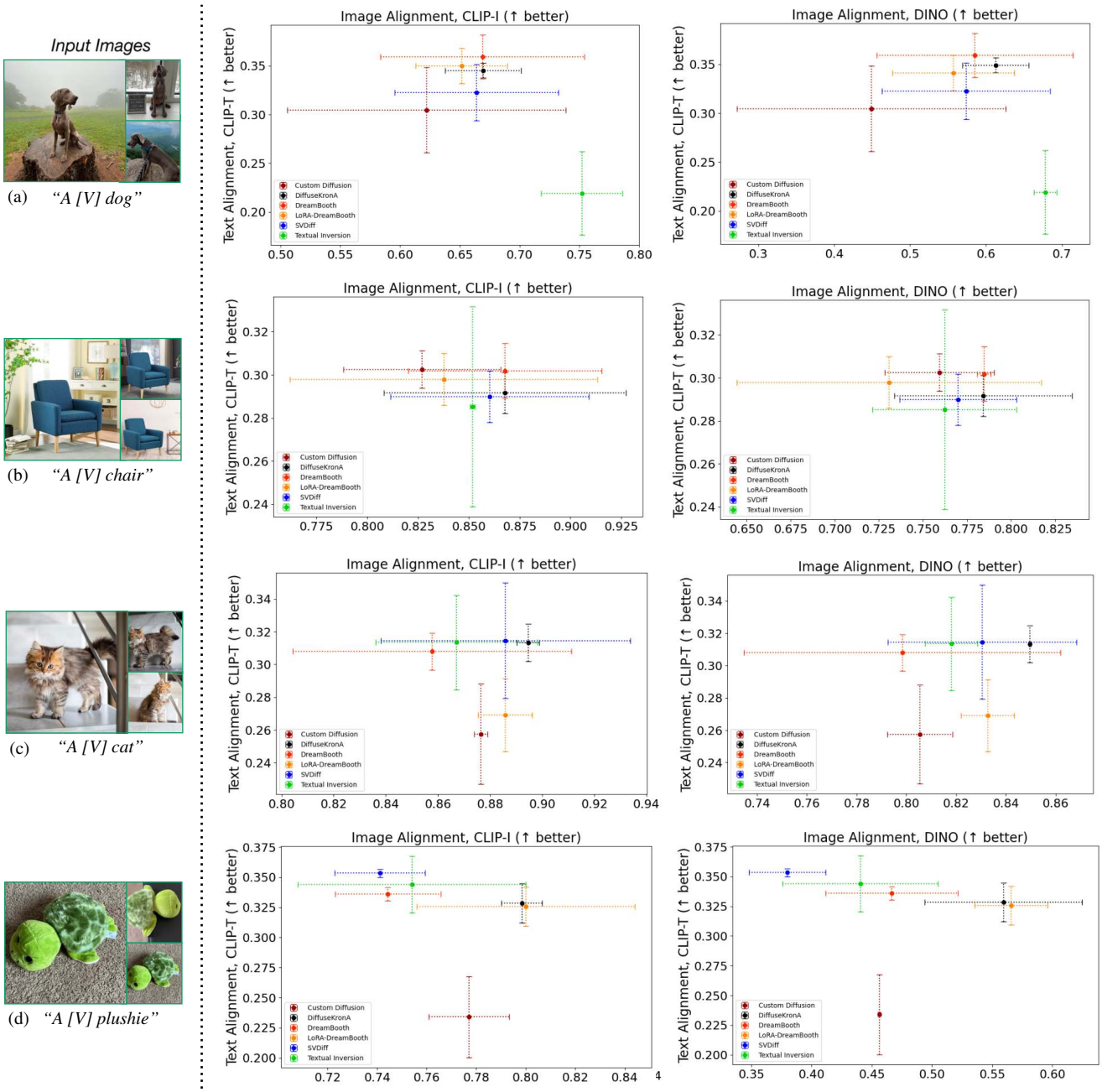


Figure 23. **Quantitative comparison** of *DiffuseKronA* with SOTA on Text-Image Alignment. The scores are computed from the same set of images and prompts as depicted in Fig. 22.

References

[1] Wenhu Chen, Hexiang Hu, Yandong Li, Nataniel Ruiz, Xuhui Jia, Ming-Wei Chang, and William W Cohen. Subject-driven text-to-image generation via apprenticeship learning. *Advances in Neural Information Processing Systems*, 36, 2024. 29

[2] CompVis. stable-diffusion, 2021. 30

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 1

[4] Ali Edalati, Marzieh Tahaei, Ivan Kobyzev, Vahid Partovi Nia, James J Clark, and Mehdi Rezagholizadeh. Krona: Parameter efficient tuning with kronecker adapter. *arXiv preprint arXiv:2212.10650*, 2022. 1

[5] Ali Edalati, Marzieh Tahaei, Ahmad Rashid, Vahid Nia, James Clark, and Mehdi Rezagholizadeh. Kronecker de-

composition for gpt compression. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, page 219–226. Association for Computational Linguistics, 2022. 3

[6] Ali Edalati, Marzieh S Tahaei, Ahmad Rashid, Vahid Partovi Nia, James J Clark, and Mehdi Rezagholizadeh. Compacter: Efficient low-rank hypercomplex adapter layers. *arXiv preprint arXiv:2106.04647*, 2021. 1

[7] 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, 2022. 29

[8] Marawan Gamal Abdel Hameed, Marzieh S Tahaei, Ali Mosleh, Vahid Partovi Nia, Hengnu Chen, Lei Deng, Tianyi Yan, and Guoqi Li. Convolutional neural network compression through generalized kronecker product decomposition. *IEEE Transactions on Neural Networks and Learning Systems*, 34(5):2205–2219, 2023. 1

[9] Ligong Han, Yinxiao Li, Han Zhang, Peyman Milanfar, Dimitris Metaxas, and Feng Yang. Svdiff: Compact parameter space for diffusion fine-tuning. *arXiv preprint arXiv:2303.11305*, 2023. 29

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, 2015. 6

[11] Xuehai He, Chunyuan Li, Pengchuan Zhang, Jianwei Yang, and Xin Eric Wang. Parameter-efficient model adaptation for vision transformers. *arXiv preprint arXiv:2203.16329*, 2022. 1

[12] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *Int. Conf. Mach. Learn.*, pages 2790–2799. PMLR, 2019. 1

[13] Nupur Kumari, Bingliang Zhang, Richard Zhang, Eli Shechtman, and Jun-Yan Zhu. Multi-concept customization of text-to-image diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1931–1941, 2023. 4, 29

[14] Chunyuan Li, Heerad Farkhoor, Rosanne Liu, and Jason Yosinski. Measuring the intrinsic dimension of objective landscapes. In *International Conference on Learning Representations*, 2018. 1

[15] Dongxu Li, Junnan Li, and Steven CH Hoi. Blip-diffusion: Pre-trained subject representation for controllable text-to-image generation and editing. *arXiv preprint arXiv:2305.14720*, 2023. 29

[16] Jian Ma, Junhao Liang, Chen Chen, and Haonan Lu. Subject-diffusion: Open domain personalized text-to-image generation without test-time fine-tuning. *arXiv preprint arXiv:2307.11410*, 2023. 29

[17] James G Nagy and Lisa Perrone. Kronecker products in image restoration. In *Advanced Signal Processing Algorithms, Architectures, and Implementations XIII*, volume 5205, page 155–163. International Society for Optics and Photonics, 2003. 1

[18] Xichen Pan, Li Dong, Shaohan Huang, Zhiliang Peng, Wenhu Chen, and Furu Wei. Kosmos-g: Generating images in context with multimodal large language models. *arXiv preprint arXiv:2310.02992*, 2023. 29

[19] 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, 2023. 15

[20] Zeju Qiu, Weiyang Liu, Haiwen Feng, Yuxuan Xue, Yao Feng, Zhen Liu, Dan Zhang, Adrian Weller, and Bernhard Schölkopf. Controlling text-to-image diffusion by orthogonal finetuning. *Advances in Neural Information Processing Systems*, 36:79320–79362, 2023. 29

[21] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551, 2020. 1

[22] 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, June 2023. 4, 29

[23] Simo Ryu. Low-rank adaptation for fast text-to-image diffusion fine-tuning, 2023. 29

[24] Marzieh Tahaei, Ella Charlaix, Vahid Nia, Ali Ghodsi, and Mehdi Rezagholizadeh. KroneckerBERT: Significant compression of pre-trained language models through kronecker decomposition and knowledge distillation. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, page 2116–2127, Seattle, United States, 2022. Association for Computational Linguistics. 1

[25] Urmish Thakker, Jesse Beu, Dibakar Gope, Chu Zhou, Igor Fedorov, Ganesh Dasika, and Matthew Mattina. Compressing rnns for iot devices by 15-38x using kronecker products. *arXiv preprint arXiv:1906.02876*, 2019. 1, 3

[26] Dingheng Wang, Bijiao Wu, Guangshe Zhao, Man Yao, Hengnu Chen, Lei Deng, Tianyi Yan, and Guoqi Li. Kronecker cp decomposition with fast multiplication for compressing rnns. *IEEE Transactions on Neural Networks and Learning Systems*, 34(5):2205–2219, 2023. 3

[27] Shin-Ying Yeh, Yu-Guan Hsieh, Zhidong Gao, Bernard BW Yang, Giyeong Oh, and Yanmin Gong. Navigating text-to-image customization: From lycoris fine-tuning to model evaluation. *arXiv preprint arXiv:2309.14859*, 2023. 15, 17, 18

[28] Aston Zhang, Yi Tay, SHUAI Zhang, Alvin Chan, Anh Tuan Luu, Siu Hui, and Jie Fu. Beyond fully-connected layers with quaternions: Parameterization of hypercomplex multiplications with $1/n$ parameters. In *International Conference on Learning Representations*, 2020. 1

[29] Yufan Zhou, Ruiyi Zhang, Jiuxiang Gu, and Tong Sun. Customization assistant for text-to-image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9182–9191, 2024. 29