# *Supplementary Materials for* DreamView: Injecting View-specific Text Guidance into Text-to-3D Generation

In this supplementary material, we provide the details of the dataset we rendered and our implementations (Section 1). Additionally, we present more qualitative text-to-3D generation results in Section 2, including more comparisons, more results with general prompts, and combining DreamView-2D with another 2D-lifting technique. And we also discuss directly applying our injection module to other text-to-image models for 3D generation and some other ways to exploit the view-specific guidance. Besides, we discuss two potential ways to reduce the users' burden to write view-specific prompts in Section 3. The dataset, model, and code will be released at here.

## 1 More Implementation Details

**Rendering Details.** Our dataset is rendered from the Objaverse [3] dataset, which contains more than 800k 3D assets. We follow the following pipeline to construct our dataset.

1. **Rendering**: For each 3D asset, we first normalize the object at the center covered by a bounding box between $[-0.5, 0.5]$. Then, we random sample a camera elevation between $[0, 30]$ and adjust the camera distance to ensure the entire 3D object is visible at any azimuth. Last, we uniformly render 32 images for the 3D object, starting from azimuth=0, and the camera position is saved at the same time. The illuminant comes from a three-point light source. The rendered size is set to be $512 \times 512$, and the background of the rendered image is a gray background. In total, we render ∼435k 3D assets, resulting in ∼14M rendered images.
2. **Captioning**: We apply the BLIP-2 [5] captioning model to caption the rendered images. For each image, we generate 5 captions. Then, we use CLIP [11] to select one caption with the highest CLIP score with the rendered image, forming the view-specific text.
3. **Merging**: We use GPT-4 [9] to merge the 32 view-specific text, forming the overall text. The input of GPT-4 is a sentence templated as "*Given a set of descriptions about the same 3D object, distill these descriptions into one concise caption. The descriptions are as follows: 'text'. Avoid describing the background. The caption should be:*", where the 'text' is the used 32 texts.

We show some samples from our dataset in Figure 1. As we can see, the view-specific text does describe the content visible from its corresponding viewpoint. For example, only the cape (robe) can be seen in the sample's first and fourth image views in the top-right corner, so their texts only describe the cape (robe,


**Fig. 1:** Samples from our dataset. We render 32 views for each 3D object in the Objaverse [3] dataset and 4 orthogonal image views are shown in this figure. The text beside the image is its corresponding view-specific text and the text below the image is the overall text.

bag). Moreover, the overall text integrates text descriptions from other viewpoints, thus providing a more comprehensive description, *e.g.*, the "Batman" and "from Five Nights at Freddy's" in the top-right and bottom-right samples.

**DreamView-2D.**

1. **Training**: We use a combined dataset consisting of our rendered dataset and a subset of the 2D LAION dataset [13]. In each training iteration, we select 3D data with a 70% probability and 2D data with a 30% probability. For each 3D object, we randomly select 4 orthogonal image views, and their captions are suffixed with "3D asset" if the raw captions do not contain the word '3D' to distinguish from 2D data. We use 16 V100 to train the model. The batch size is 256 (4 objects, each object has 4 image views, across 16 GPUs), and the gradient is accumulated from 8 batches, forming a total batch size of 2,048. We use the AdamW [8] optimizer, and the learning rate is set to $1e^{-4}$. The image size is $256 \times 256$. We randomly drop the text condition with a

10% probability. The margin is randomly sampled from -0.1 to 0.1 for each training iteration. We use SD-v2.1 as the model initialization, and the base network architecture is mostly the same as those used in [10, 14, 16, 17].

2. **Inference**: We adopt the DDIM [15] sampler with 50 sampling steps and a classifier-free guidance (CFG) scale of 7.5 for the text-to-image generation. By default, we generate 4 views simultaneously in inference and adopt a margin of -0.025.

**DreamView-3D.** We implement DreamView-3D by building upon threestudio [4] and substituting Stable Diffusion [12] in DreamFusion [10] with our text-to-image model (DreamView-2D) for text-to-3D generation. To represent the 3D content, we employ the implicit-volume approach (Mip-Nerf [1]) and optimize it for 10,000 steps using the AdamW optimizer [8] with a learning rate of 0.01. The optimization takes about 55 minutes on an A100 GPU. During the first 8,000 optimization steps, SDS's maximum and minimum time steps are linearly decreased from 0.98 to 0.5 and 0.02, respectively. The CFG scale is set to 50, and we use a rescaling factor of 0.5 for the CFG rescaling. Initially, the rendering resolution and batch size are set to $64 \times 64$ and 8 for the first 5,000 steps and are then changed to $256 \times 256$ and 4. After 5,000 steps, we enable soft shading [6]. In most cases, we divide the azimuth angle of the camera position into four intervals: $[10, 170]$ is the front side, $(170, 190)$ is the right side, $[190, 350]$ is the back side, and the remaining part is the left side. We use "*ugly, bad anatomy, blurry, pixelated obscure, unnatural colors, poor lighting, dull, and unclear, cropped, lowres, low quality, artifacts, duplicate, morbid, mutilated, poorly drawn face, deformed, dehydrated, bad proportions*" as the negative text prompt. The margin of the DreamView-2D is set to be -0.025.

## 2 More Evaluations

### 2.1 Comparisons with Others on Text-to-3D Generation

In Figures 2 to 4, we compared the 3D generation results of 7 different methods under 3 text prompts containing customized content. We start from azimuth=0 degrees and show a total of 8 views at 45-degree intervals. These results demonstrate that other methods either suffer from the 3D inconsistent problem, *e.g.*, multi-face and multi-foot problems, or cannot accurately generate specified content based on the text. In contrast, our method can include both 3D consistency and viewpoint customization capabilities.

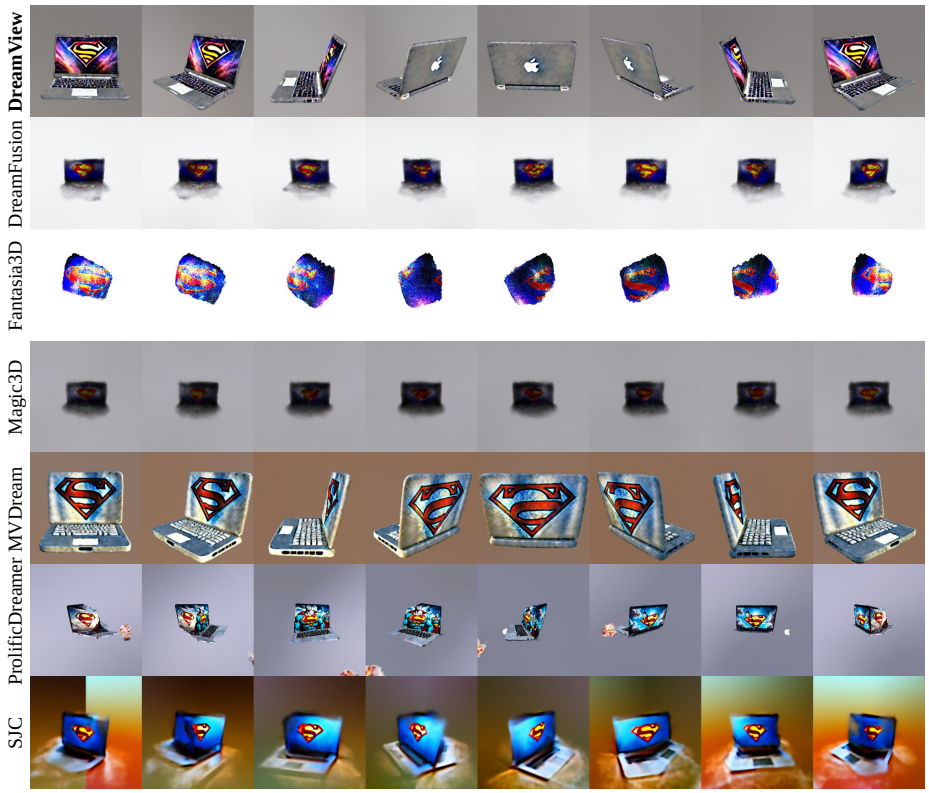


**Fig. 2:** 3D generation results of "An open MAC book showing the logo of Superman on the screen". View-specific text: -Front: "An open MAC book showing the logo of Superman on the screen". -Back: "Back and side view of an open MAC book with an apple logo". -Side: "Side view of an open MAC book". Note that the view-specific texts for the right and left are the same.


**Fig. 3:** 3D generation results of "A penguin wearing a scarf, carrying a crossbody bag on the back, 8K, HD, 3d render, best quality". View-specific text: -Front: "A penguin wearing a scarf, 8K, HD, 3d render, best quality". -Back: "A penguin carrying a crossbody bag on the back, 8K, HD, 3d render, best quality". -Side: "A penguin wearing a scarf, carrying a crossbody bag on the back, 8K, HD, 3d render, best quality". Note that the view-specific texts for the right and left are the same.


**Fig. 4:** 3D generation results of "An astronaut riding a horse, wearing a white space suit, carrying a red backpack on the back". View-specific text: -Front: "A horse and an astronaut with a white space suit is riding the horse". -Back: "A red backpack is carried on the astronaut's back". -Side: "A red backpack, a horse, an astronaut with a white space suit". Note that the view-specific texts for the right and left are the same.


**Fig. 5:** Comparisons between DreamView-3D and image-to-3D methods. The reference image for performing image-to-3D is generated by SD-XL with the overall text.

### 2.2 Discussion with Image-to-3D Generation

In addition to text-to-3D generation, another stream of methods synthesizes 3D assets via lifting 2D images. We compare ours with them in Figure 5, where the reference image is generated by SD-XL using the overall text prompt. According to the results, although SD-XL can generate the concepts of the penguin, scarf, and crossbody bag, the crossbody bag is still overlooked when the generated image is lifted to a 3D object via Zero-1-to-3 XL [2] and SyncDreamer [7]. More importantly, these methods accept a single image for generating 3D assets, which inherently contradicts the diversity of appearance across viewpoints, making such singe-image-to-3D methods difficult to achieve customizable 3D generation.


**Fig. 6:** 3D generation results of our DreamView with general prompts, *i.e.*, only the overall text is used.

### 2.3 More Results with General Prompts

We also show more generation results with general prompts in Figure 6, where the used overall text prompts are from the prompt library [1]. The results demonstrate that our model can also work with general prompts and generate highly consistent 3D objects with high fidelity.

[1] Prompt Library

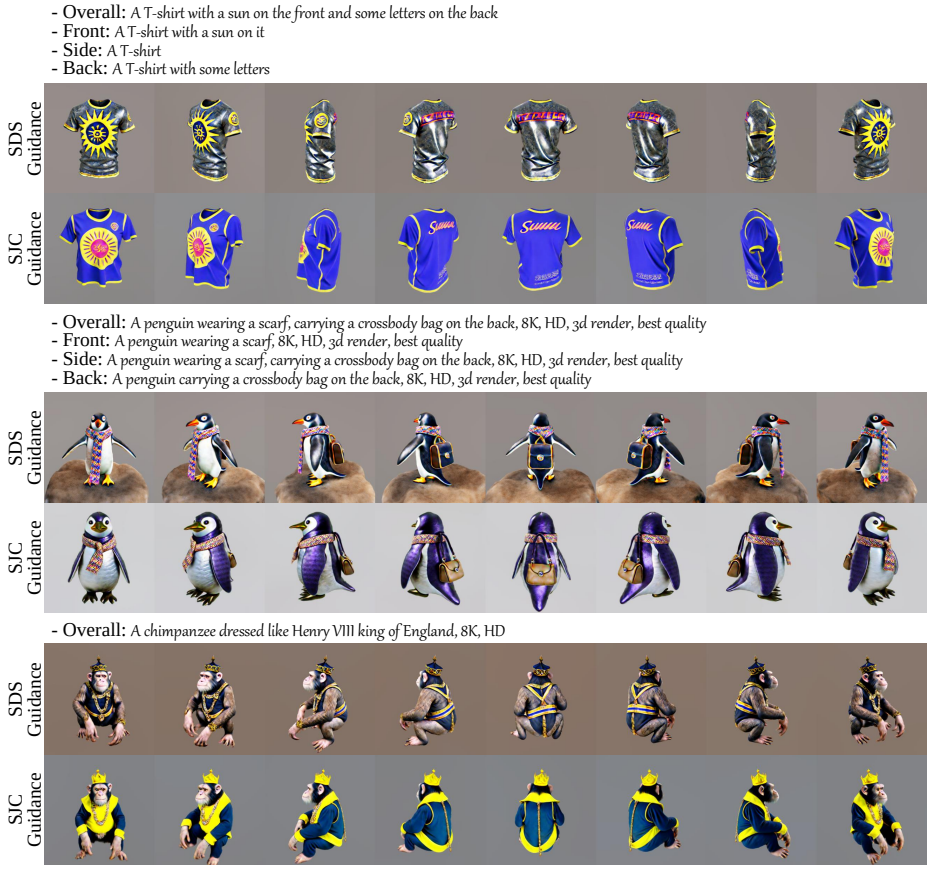


**Fig. 7:** 3D generation results via SDS guidance [10] and SJC guidance [16]. The first two cases adopt view-specific texts, and the last case merely uses the overall text.

### 2.4 Combing DreamView-2D with SJC Guidance

In the main manuscript, we mainly build our DreamView-3D upon the SDS guidance proposed in DreamFusion [10]. Now, we adopt the SJC [16] guidance to lift our DreamView-2D to conduct text-to-3D generation, where the results are shown in Figure 7. The results demonstrate that DreamView-2D can work with other 2D-lifting techniques, such as SJC, not limited to SDS, and the properties of customization and consistency are still preserved. Besides, DreamView-2D is expected to be combined with VSD [17] to further improve the generation quality, which will be reserved for future work.

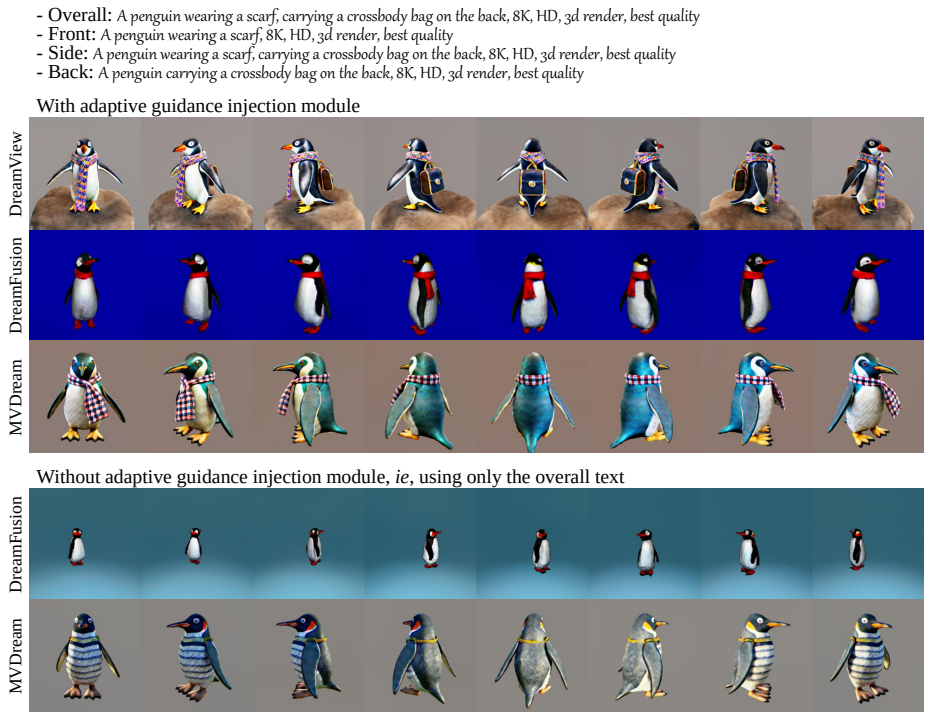


**Fig. 8:** 3D generation results of directly applying our adaptive injection module to DreamFusion and MVDream. The scarf is presented correctly for DreamFusion and MVDream with the injection module, but they still fail to generate the crossbody bag.

### 2.5 Applying the Injection Module to Other Text-to-image Models

We directly apply our proposed adaptive guidance injection module to the text-to-image models used in DreamFusion [10] and MVDream [14] to verify if the injection module can introduce customization ability to them.

The results in Figure 8 show that when the injection module is directly applied to DreamFusion, the scarf can be successfully presented while still failing to generate the crossbody bag. A similar phenomenon also occurs when applying the injection module to MVDream. Overall, our injection module can be directly integrated with these models and enables them to achieve some viewpoint customization without view customization training, verifying the universality of the injection method. However, their customization capability is not as ideal as our DreamView.


**Fig. 9:** Comparisons between adaptive injection and another two injection methods.

### 2.6 Other Ways to Use View-specific Text Guidance

We propose an adaptive way to use the view-specific and overall text guidance in DreamView. Except for adaptively injecting view-specific text guidance, we explore two other ways to inject them, termed **concatenating** and **alternating** injection. The former concatenates the text embeddings of the overall text and view-specific text to form the condition for all diffusion U-Net blocks. The latter alternately injects the overall text and view-specific text into the U-Net block. Specifically, it injects the view-specific text embedding to a U-Net block whose Layer ID can not be divided by 2. Otherwise, the overall text embedding is injected. In brief, these two injection methods inject the overall and view-specific text in a pre-defined static way. Compared with them, the adaptive injection module used in DreamView determines which guidance should be used in the current U-Net block according to the similarities between the text-image embedding, which is a dynamic process.

The comparisons are shown in Figure 9. According to the results, concatenating injection fails to fully customize the 3D content, *e.g.*, the crossbody bag and the hammer. Besides, the alternating injection performs well in generating the penguin. However, it fails to generate the hammer in the second case, where we also observe slight 3D inconsistency, *i.e.*, the multi-face problem and some noise.

In conclusion, our adaptive injection module, which utilizes two kinds of guidance in a dynamic way, can achieve a better balance between consistency and customization than the concatenating and alternating injection methods that inject the overall and view-specific text in a static way.

## 3 Reducing the Burden of Writing Prompts

All the text prompts used in DreamView, including the overall text and four view-specific prompts, are hand-written by default. Although such a way can ensure the accuracy of the description as much as possible, it undoubtedly brings additional burdens to users. Here, we discuss two potential ways to help users lower the burden of providing view-specific text prompts: (1) Appending the direction description text to the overall text to generate view-specific texts; (2) Asking Large Language Models (LLMs) for generating view-specific texts. We take the "penguin" case as an example and discuss them as follows.

**Way 1:** We append the direction description text, *e.g.*, 'from the front view' and 'from the back view', *etc.*, to the overall text to generate view-specific text, which is inspired by the view-dependent conditioning in DreamFusion [10] that initially designed for alleviating the 3D inconsistent problem. The results are shown in the second row of Figure 10, where the crossbody bag is not successfully presented. We conjecture this is because the model has rarely learned the concept of direction description during its training. A possible solution is to add direction descriptions to the texts in the training set and retrain the model, but this may incur some annotation costs, which will be further explored in the future.

**Way 2:** We ask GPT-4 [9] to generate view-specific texts in a one-shot way, where the used text prompt template is:

*The input is an overall text description of a 3D object from the global level. Please output its descriptions from the front, right, back, and left view based on this overall description, and return the result in a JSON format. I will give you an example:*

*Input: 'An Iron man carrying the shield of Captain America on the back'.*

*Output: {'Front': 'An Iron man', 'Right': 'An Iron man with a shield on the back', 'Back': 'The shield of Captain America is on the Iron man's back', 'Left': 'An Iron man with a shield on the back'}*

*Now, what is the output of 'A Penguin wearing a scarf, carrying a crossbody bag on the back, 8K, HD, 3D render, best quality'? Note: do not delete the text describing quality, eg, 8K, HD, and so on.*

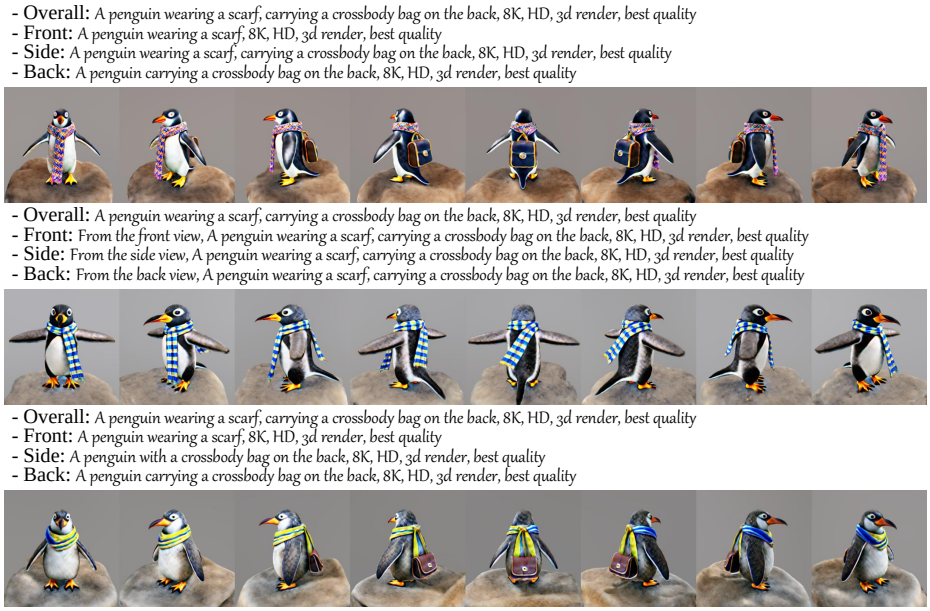


**Fig. 10:** First row: 3D generation results of hand-written text prompts. Second row: 3D generation results of direction description text prompts. Last row: 3D generation results of text prompts given by GPT-4 [9].

**Table 1:** Comparisons on hand-written and GPT-given prompts.

| How to obtain | CLIP Score (↑) | | Inception |
|---|---|---|---|
| View-specific text | Overall Text | View Text | Score (↑) |
| Hand-written | 31.1 | 32.1 | 14.5 |
| GPT3.5-given | 31.4 | 31.6 | 14.8 |

The output text prompts and the 3D generation results are shown in the last row of Figure 10. The text prompts given by GPT-4 can successfully generate the penguin with a scarf and a crossbody bag, suggesting that using LLMs to generate view-specific texts may become a potential way to reduce user burden. Besides, Table 1 is the quantitative result of text-to-image generation on the validation set between hand-written and GPT3.5-given prompts, showing that the GPT-given prompts are quantitatively on par with hand-written prompts.

**A quick verification of prompts.** We here provide a quick verification of whether the used prompts can achieve satisfactory customization. In practice, users can refine their prompts according to the text-to-image generation results, as the quality of 3D generation is highly correlated with image generation. Since the image generation is much faster than the 3D generation (3 seconds *vs.* 55 minutes), the prompt refinement process will not take long.

## References


1. Barron, J.T., Mildenhall, B., Tancik, M., Hedman, P., Martin-Brualla, R., Srinivasan, P.P.: Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In: ICCV (2021)
2. Deitke, M., Liu, R., Wallingford, M., Ngo, H., Michel, O., Kusupati, A., Fan, A., Laforte, C., Voleti, V., Gadre, S.Y., et al.: Objaverse-xl: A universe of 10m+ 3d objects. In: NeurIPS (2024)
3. Deitke, M., Schwenk, D., Salvador, J., Weihs, L., Michel, O., VanderBilt, E., Schmidt, L., Ehsani, K., Kembhavi, A., Farhadi, A.: Objaverse: A universe of annotated 3d objects. In: CVPR (2023)
4. Guo, Y.C., Liu, Y.T., Shao, R., Laforte, C., Voleti, V., Luo, G., Chen, C.H., Zou, Z.X., Wang, C., Cao, Y.P., Zhang, S.H.: threestudio: A unified framework for 3d content generation. `https://github.com/threestudio-project/threestudio` (2023)
5. Li, J., Li, D., Savarese, S., Hoi, S.: Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In: ICML (2023)
6. Lin, C.H., Gao, J., Tang, L., Takikawa, T., Zeng, X., Huang, X., Kreis, K., Fidler, S., Liu, M.Y., Lin, T.Y.: Magic3d: High-resolution text-to-3d content creation. In: CVPR (2023)
7. Liu, Y., Lin, C., Zeng, Z., Long, X., Liu, L., Komura, T., Wang, W.: Syncdreamer: Generating multiview-consistent images from a single-view image. In: ICLR (2024)
8. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. In: ICLR (2018)
9. OpenAI: Gpt-4 technical report. arXiv (2023)
10. Poole, B., Jain, A., Barron, J.T., Mildenhall, B.: Dreamfusion: Text-to-3d using 2d diffusion. In: ICLR (2023)
11. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al.: Learning transferable visual models from natural language supervision. In: ICML (2021)
12. Rombach, R., Blattmann, A., Lorenz, D., Esser, P., Ommer, B.: High-resolution image synthesis with latent diffusion models. In: CVPR (2022)
13. Schuhmann, C., Beaumont, R., Vencu, R., Gordon, C., Wightman, R., Cherti, M., Coombes, T., Katta, A., Mullis, C., Wortsman, M., et al.: Laion-5b: An open large-scale dataset for training next generation image-text models. In: NeurIPS (2022)
14. Shi, Y., Wang, P., Ye, J., Mai, L., Li, K., Yang, X.: Mvdream: Multi-view diffusion for 3d generation. In: ICLR (2024)
15. Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., Ganguli, S.: Deep unsupervised learning using nonequilibrium thermodynamics. In: ICML (2015)
16. Wang, H., Du, X., Li, J., Yeh, R.A., Shakhnarovich, G.: Score jacobian chaining: Lifting pretrained 2d diffusion models for 3d generation. In: CVPR (2023)
17. Wang, Z., Lu, C., Wang, Y., Bao, F., Li, C., Su, H., Zhu, J.: Prolificdreamer: High-fidelity and diverse text-to-3d generation with variational score distillation. In: NeurIPS (2023)