

---

# SDXL: Improving Latent Diffusion Models for High-Resolution Image Synthesis

---

Dustin Podell   Zion English   Kyle Lacey   Andreas Blattmann   Tim Dockhorn

Jonas Müller

Joe Penna

Robin Rombach

Stability AI, Applied Research

Code: <https://github.com/Stability-AI/generative-models>   Model weights: <https://huggingface.co/stabilityai/>



## Abstract

We present *SDXL*, a latent diffusion model for text-to-image synthesis. Compared to previous versions of *Stable Diffusion*, *SDXL* leverages a three times larger UNet backbone: The increase of model parameters is mainly due to more attention blocks and a larger cross-attention context as *SDXL* uses a second text encoder. We design multiple novel conditioning schemes and train *SDXL* on multiple aspect ratios. We also introduce a *refinement model* which is used to improve the visual fidelity of samples generated by *SDXL* using a post-hoc *image-to-image* technique. We demonstrate that *SDXL* shows drastically improved performance compared to previous versions of *Stable Diffusion* and achieves results competitive with those of black-box state-of-the-art image generators. In the spirit of promoting open research and fostering transparency in large model training and evaluation, we provide access to code and model weights.

# 1 Introduction

The last year has brought enormous leaps in deep generative modeling across various data domains, such as natural language [50], audio [17], and visual media [38, 37, 40, 44, 15, 3, 7]. In this report, we focus on the latter and unveil *SDXL*, a drastically improved version of *Stable Diffusion*. *Stable Diffusion* is a latent text-to-image diffusion model (DM) which serves as the foundation for an array of recent advancements in, e.g., 3D classification [43], controllable image editing [54], image personalization [10], synthetic data augmentation [48], graphical user interface prototyping [51], etc. Remarkably, the scope of applications has been extraordinarily extensive, encompassing fields as diverse as music generation [9] and reconstructing images from fMRI brain scans [49].

User studies demonstrate that *SDXL* consistently surpasses all previous versions of *Stable Diffusion* by a significant margin (see Fig. 1). In this report, we present the design choices which lead to this boost in performance encompassing *i*) a  $3\times$  larger UNet-backbone compared to previous *Stable Diffusion* models (Sec. 2.1), *ii*) two simple yet effective additional conditioning techniques (Sec. 2.2) which do not require any form of additional supervision, and *iii*) a separate diffusion-based refinement model which applies a noising-denoising process [28] to the latents produced by *SDXL* to improve the visual quality of its samples (Sec. 2.5).

A major concern in the field of visual media creation is that while black-box-models are often recognized as state-of-the-art, the opacity of their architecture prevents faithfully assessing and validating their performance. This lack of transparency hampers reproducibility, stifles innovation, and prevents the community from building upon these models to further the progress of science and art. Moreover, these closed-source strategies make it challenging to assess the biases and limitations of these models in an impartial and objective way, which is crucial for their responsible and ethical deployment. With *SDXL* we are releasing an *open* model that achieves competitive performance with black-box image generation models (see Fig. 10 & Fig. 11).

## 2 Improving *Stable Diffusion*

In this section we present our improvements for the *Stable Diffusion* architecture. These are modular, and can be used individually or together to extend any model. Although the following strategies are implemented as extensions to latent diffusion models (LDMs) [38], most of them are also applicable to their pixel-space counterparts.

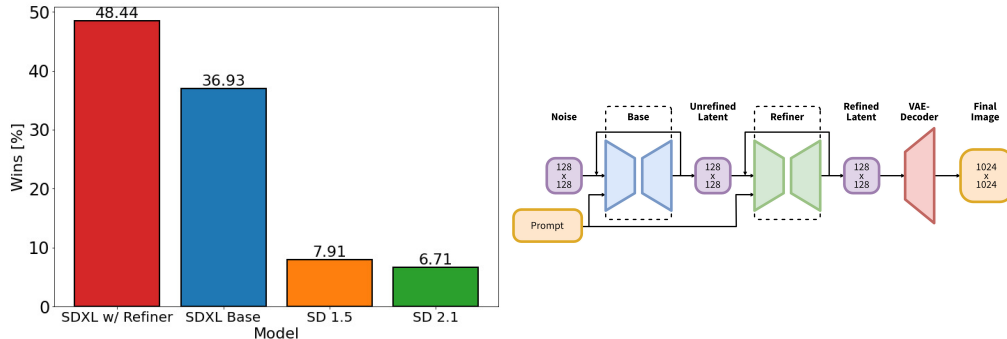


Figure 1: *Left*: Comparing user preferences between *SDXL* and *Stable Diffusion* 1.5 & 2.1. While *SDXL* already clearly outperforms *Stable Diffusion* 1.5 & 2.1, adding the additional refinement stage boosts performance. *Right*: Visualization of the two-stage pipeline: We generate initial latents of size  $128 \times 128$  using *SDXL*. Afterwards, we utilize a specialized high-resolution *refinement model* and apply SDEdit [28] on the latents generated in the first step, using the same prompt. *SDXL* and the refinement model use the same autoencoder.

### 2.1 Architecture & Scale

Starting with the seminal works Ho et al. [14] and Song et al. [47], which demonstrated that DMs are powerful generative models for image synthesis, the convolutional UNet [39] architecture has been the dominant architecture for diffusion-based image synthesis. However, with the development

Table 1: Comparison of *SDXL* and older *Stable Diffusion* models.

Model	<i>SDXL</i>	SD 1.4/1.5	SD 2.0/2.1
# of UNet params	2.6B	860M	865M
Transformer blocks	[0, 2, 10]	[1, 1, 1, 1]	[1, 1, 1, 1]
Channel mult.	[1, 2, 4]	[1, 2, 4, 4]	[1, 2, 4, 4]
Text encoder	CLIP ViT-L & OpenCLIP ViT-bigG	CLIP ViT-L	OpenCLIP ViT-H
Context dim.	2048	768	1024
Pooled text emb.	OpenCLIP ViT-bigG	N/A	N/A

of foundational DMs [40, 37, 38], the underlying architecture has constantly evolved: from adding self-attention and improved upscaling layers [5], over cross-attention for text-to-image synthesis [38], to pure transformer-based architectures [33].

We follow this trend and, following Hooeboom et al. [16], shift the bulk of the transformer computation to lower-level features in the UNet. In particular, and in contrast to the original *Stable Diffusion* architecture, we use a heterogeneous distribution of transformer blocks within the UNet: For efficiency reasons, we omit the transformer block at the highest feature level, use 2 and 10 blocks at the lower levels, and remove the lowest level ( $8\times$  downsampling) in the UNet altogether — see Tab. 1 for a comparison between the architectures of *Stable Diffusion* 1.x & 2.x and *SDXL*. We opt for a more powerful pre-trained text encoder that we use for text conditioning. Specifically, we use OpenCLIP ViT-bigG [19] in combination with CLIP ViT-L [34], where we concatenate the penultimate text encoder outputs along the channel-axis [1]. Besides using cross-attention layers to condition the model on the text-input, we follow [30] and additionally condition the model on the pooled text embedding from the OpenCLIP model. These changes result in a model size of 2.6B parameters in the UNet, see Tab. 1. The text encoders have a total size of 817M parameters.

## 2.2 Micro-Conditioning

**Conditioning the Model on Image Size** A notorious shortcoming of the LDM paradigm [38] is the fact that training a model requires a *minimal image size*, due to its two-stage architecture. The two main approaches to tackle this problem are either to discard all training images below a certain minimal resolution (for example, *Stable Diffusion* 1.4/1.5 discarded all images with any size below 512 pixels), or, alternatively, upscale images that are too small. However, depending on the desired image resolution, the former method can lead to significant portions of the training data being discarded, what will likely lead to a loss in performance and hurt generalization. We visualize such effects in Fig. 2 for the dataset on which *SDXL* was pretrained. For this particular choice of data, discarding all samples below our pretraining resolution of  $256^2$  pixels would lead to a significant 39% of discarded data. The second method, on the other hand, usually introduces upscaling artifacts which may leak into the final model outputs, causing, for example, blurry samples.

Instead, we propose to condition the UNet model on the original image resolution, which is trivially available during training. In particular, we provide the original (i.e., before any rescaling) height and width of the images as an additional conditioning to the model  $c_{\text{size}} = (h_{\text{original}}, w_{\text{original}})$ . Each component is independently embedded using a Fourier feature encoding, and these encodings are concatenated into a single vector that we feed into the model by adding it to the timestep embedding [5].

At inference time, a user can then set the desired *apparent resolution* of the image via this *size-conditioning*. Evidently (see Fig. 3), the model has learned to associate the conditioning  $c_{\text{size}}$  with

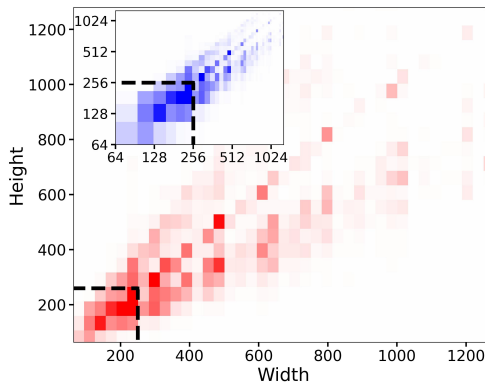


Figure 2: Height-vs-Width distribution of our pre-training dataset. Without the proposed size-conditioning, 39% of the data would be discarded due to edge lengths smaller than 256 pixels as visualized by the dashed black lines. Color intensity in each visualized cell is proportional to the number of samples.

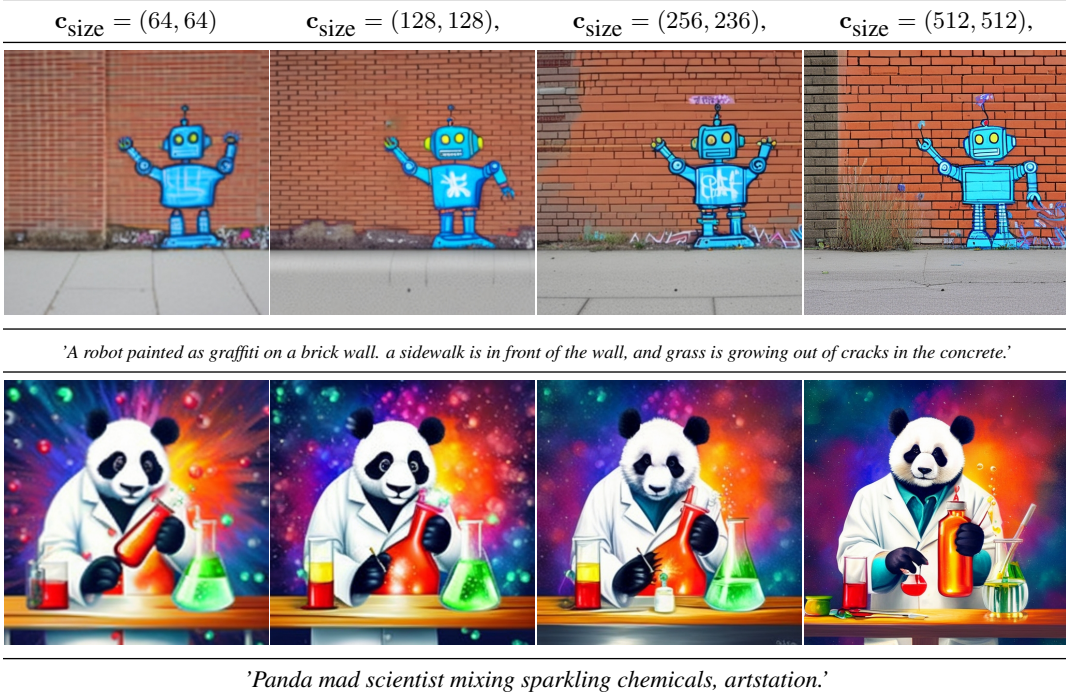


Figure 3: The effects of varying the size-conditioning: We show draw 4 samples with the same random seed from *SDXL* and vary the size-conditioning as depicted above each column. The image quality clearly increases when conditioning on larger image sizes. Samples from the  $512^2$  model, see Sec. 2.5. Note: For this visualization, we use the  $512 \times 512$  pixel base model (see Sec. 2.5), since the effect of size conditioning is more clearly visible before  $1024 \times 1024$  finetuning. Best viewed zoomed in.

resolution-dependent image features, which can be leveraged to modify the appearance of an output corresponding to a given prompt. Note that for the visualization shown in Fig. 3, we visualize samples generated by the  $512 \times 512$  model (see Sec. 2.5 for details), since the effects of the size conditioning are less clearly visible after the subsequent multi-aspect (ratio) finetuning which we use for our final *SDXL* model.

We quantitatively assess the effects of this simple but effective conditioning technique by training and evaluating three LDMs on class conditional ImageNet [4] at spatial size  $512^2$ : For the first model (*CIN-512-only*) we discard all training examples with at least one edge smaller than 512 pixels what results in a train dataset of only 70k images. For *CIN-nocond* we use all training examples but without size conditioning. This additional conditioning is only used for *CIN-size-cond*. After training we generate 5k samples with 50 DDIM steps [46] and (classifier-free) guidance scale of 5 [13] for every model and compute IS [42] and FID [12] (against the full validation set). For *CIN-size-cond* we generate samples always conditioned on  $c_{\text{size}} = (512, 512)$ . Tab. 2 summarizes the results and verifies that *CIN-size-cond* improves upon the baseline models in both metrics. We attribute the degraded performance of *CIN-512-only* to bad generalization due to overfitting on the small training dataset while the effects of a mode of blurry samples in the sample distribution of *CIN-nocond* result in a reduced FID score. Note that, although we find these classical quantitative scores not to be suitable for evaluating the performance of foundational (text-to-image) DMs [40, 37, 38] (see App. F), they remain reasonable metrics on ImageNet as the neural backbones of FID and IS have been trained on ImageNet itself.

Table 2: Conditioning on the original spatial size of the training examples improves performance on class-conditional ImageNet [4] on  $512^2$  resolution.

model	FID-5k ↓	IS-5k ↑
<i>CIN-512-only</i>	43.84	110.64
<i>CIN-nocond</i>	39.76	211.50
<i>CIN-size-cond</i>	<b>36.53</b>	<b>215.34</b>

**Conditioning the Model on Cropping Parameters** The first two rows of Fig. 4 illustrate a typical failure mode of previous *SD* models: Synthesized objects can be cropped, such as the cut-off head of the cat in the left examples for *SD* 1-5 and *SD* 2-1. An intuitive explanation for this behavior is the use of *random cropping* during training of the model: As collating a batch in DL frameworks such as



Figure 4: Comparison of the output of *SDXL* with previous versions of *Stable Diffusion*. For each prompt, we show 3 random samples of the respective model for 50 steps of the DDIM sampler [46] and cfg-scale 8.0 [13]. Additional samples in Fig. 14.

PyTorch [32] requires tensors of the same size, a typical processing pipeline is to (i) resize an image such that the shortest size matches the desired target size, followed by (ii) randomly cropping the image along the longer axis. While random cropping is a natural form of data augmentation, it can leak into the generated samples, causing the malicious effects shown above.

To fix this problem, we propose another simple yet effective conditioning method: During dataloading, we uniformly sample crop coordinates  $c_{\text{top}}$  and  $c_{\text{left}}$  (integers specifying the amount of pixels cropped from the top-left corner along the height and width axes, respectively) and feed them into the model as conditioning parameters via Fourier feature embeddings, similar to the size conditioning described above. The concatenated embedding  $\mathbf{c}_{\text{crop}}$  is then used as an additional conditioning parameter. We emphasize that this technique is not limited to LDMs and could be used for any DM. Note that crop- and size-conditioning can be readily combined. In such a case, we concatenate the feature embedding along the channel dimension, before adding it to the timestep embedding in the UNet. Alg. 1 illustrates how we sample  $\mathbf{c}_{\text{crop}}$  and  $\mathbf{c}_{\text{size}}$  during training if such a combination is applied.

Given that in our experience large scale datasets are, on average, object-centric, we set  $(c_{\text{top}}, c_{\text{left}}) = (0, 0)$  during inference and thereby obtain object-centered samples from the trained model.

See Fig. 5 for an illustration: By tuning  $(c_{\text{top}}, c_{\text{left}})$ , we can successfully *simulate* the amount of cropping during inference. This is a form of *conditioning-augmentation*, and has been used in various forms with autoregressive [20] models, and more recently with diffusion models [21].

While other methods like data bucketing [31] successfully tackle the same task, we still benefit from cropping-induced data augmentation, while making sure that it does not leak into the generation process - we actually use it to our advantage to gain more control over the image synthesis process. Furthermore, it is easy to implement and can be applied in an online fashion during training, without additional data preprocessing.

### 2.3 Multi-Aspect Training

Real-world datasets include images of widely varying sizes and aspect-ratios (c.f. fig. 2) While the common output resolutions for text-to-image models are square images of  $512 \times 512$  or  $1024 \times 1024$  pixels, we argue that this is a rather unnatural choice, given the widespread distribution and use of landscape (e.g., 16:9) or portrait format screens.

Motivated by this, we finetune our model to handle multiple aspect-ratios simultaneously: We follow common practice [31] and partition the data into buckets of different aspect ratios, where we keep the pixel count as close to  $1024^2$  pixels as possible, varying height and width accordingly in multiples

---

**Algorithm 1** Conditioning pipeline for size- and crop-conditioning

---

**Require:** Training dataset of images  $\mathcal{D}$ , target image size for training  $\mathbf{s} = (h_{\text{tgt}}, w_{\text{tgt}})$

**Require:** Resizing function  $\mathbf{R}$ , cropping function  $\mathbf{C}$

**Require:** Model train step  $\mathbf{T}$

converged  $\leftarrow$  False

**while** not converged **do**

$x \sim \mathcal{D}$

$w_{\text{original}} \leftarrow \text{width}(x)$

$h_{\text{original}} \leftarrow \text{height}(x)$

$\mathbf{c}_{\text{size}} \leftarrow (h_{\text{original}}, w_{\text{original}})$

$x \leftarrow \mathbf{R}(x, \mathbf{s})$

$\triangleright$  resize smaller image size to target size  $\mathbf{s}$

**if**  $h_{\text{original}} \leq w_{\text{original}}$  **then**

$c_{\text{left}} \sim \mathcal{U}(0, \text{width}(x) - s_w)$

$\triangleright$  sample  $c_{\text{left}}$  from discrete uniform distribution

$c_{\text{top}} = 0$

**else if**  $h_{\text{original}} > w_{\text{original}}$  **then**

$c_{\text{top}} \sim \mathcal{U}(0, \text{height}(x) - s_h)$

$\triangleright$  sample  $c_{\text{top}}$  from discrete uniform distribution

$c_{\text{left}} = 0$

**end if**

$\mathbf{c}_{\text{crop}} \leftarrow (c_{\text{top}}, c_{\text{left}})$

$x \leftarrow \mathbf{C}(x, \mathbf{s}, \mathbf{c}_{\text{crop}})$

$\triangleright$  crop image to size  $\mathbf{s}$  with top-left coordinate  $(c_{\text{top}}, c_{\text{left}})$

converged  $\leftarrow \mathbf{T}(x, \mathbf{c}_{\text{size}}, \mathbf{c}_{\text{crop}})$

$\triangleright$  train model conditioned on  $\mathbf{c}_{\text{size}}$  and  $\mathbf{c}_{\text{crop}}$

**end while**

---

$\mathbf{c}_{\text{crop}} = (0, 0)$

$\mathbf{c}_{\text{crop}} = (0, 256)$ ,

$\mathbf{c}_{\text{crop}} = (256, 0)$ ,

$\mathbf{c}_{\text{crop}} = (512, 512)$ ,



*'An astronaut riding a pig, highly realistic dslr photo, cinematic shot.'*



*'A capybara made of lego sitting in a realistic, natural field.'*

Figure 5: Varying the crop conditioning as discussed in Sec. 2.2. See Fig. 4 and Fig. 14 for samples from  $SD$  1.5 and  $SD$  2.1 which provide no explicit control of this parameter and thus introduce cropping artifacts. Samples from the 512<sup>2</sup> model, see Sec. 2.5.

of 64. A full list of all aspect ratios used for training is provided in App. I. During optimization, a training batch is composed of images from the same bucket, and we alternate between bucket sizes for each training step. Additionally, the model receives the bucket size (or, *target size*) as a conditioning, represented as a tuple of integers  $\mathbf{c}_{\text{ar}} = (h_{\text{tgt}}, w_{\text{tgt}})$  which are embedded into a Fourier space in analogy to the size- and crop-conditionings described above.

In practice, we apply multi-aspect training as a finetuning stage after pretraining the model at a fixed aspect-ratio and resolution and combine it with the conditioning techniques introduced in Sec. 2.2 via concatenation along the channel axis. Fig. 16 in App. J provides python-code for this operation. Note that crop-conditioning and multi-aspect training are complementary operations, and crop-conditioning then only works within the bucket boundaries (usually 64 pixels). For ease of implementation, however, we opt to keep this control parameter for multi-aspect models.

## 2.4 Improved Autoencoder

*Stable Diffusion* is a LDM, operating in a pre-trained, learned (and fixed) latent space of an autoencoder. While the bulk of the semantic composition is done by the LDM [38], we can improve *local*, high-frequency details in generated images by improving the autoencoder. To this end, we train the same autoencoder architecture used for the original *Stable Diffusion* at a larger batch-size (256 vs 9) and additionally track the weights with an exponential moving average. The resulting autoencoder outperforms the original model in all evaluated reconstruction metrics, see Tab. 3. We use this autoencoder for all of our experiments.

Table 3: Autoencoder reconstruction performance on the COCO2017 [26] validation split, images of size  $256 \times 256$  pixels. Note: *Stable Diffusion* 2.x uses an improved version of *Stable Diffusion* 1.x’s autoencoder, where the decoder was finetuned with a reduced weight on the perceptual loss [55], and used more compute. Note that our new autoencoder is trained from scratch.

model	PNSR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	rFID $\downarrow$
<i>SDXL</i> -VAE	<b>24.7</b>	<b>0.73</b>	<b>0.88</b>	<b>4.4</b>
<i>SD</i> -VAE 1.x	23.4	0.69	0.96	5.0
<i>SD</i> -VAE 2.x	24.5	0.71	0.92	4.7

## 2.5 Putting Everything Together

We train the final model, *SDXL*, in a multi-stage procedure. *SDXL* uses the autoencoder from Sec. 2.4 and a discrete-time diffusion schedule [14, 45] with 1000 steps. First, we pretrain a base model (see Tab. 1) on an internal dataset whose height- and width-distribution is visualized in Fig. 2 for 600 000 optimization steps at a resolution of  $256 \times 256$  pixels and a batch-size of 2048, using size- and crop-conditioning as described in Sec. 2.2. We continue training on  $512 \times 512$  pixel images for another 200 000 optimization steps, and finally utilize multi-aspect training (Sec. 2.3) in combination with an offset-noise [11, 25] level of 0.05 to train the model on different aspect ratios (Sec. 2.3, App. I) of  $\sim 1024 \times 1024$  pixel area.

**Refinement Stage** Empirically, we find that the resulting model sometimes yields samples of low local quality, see Fig. 6. To improve sample quality, we train a separate LDM in the same latent space, which is specialized on high-quality, high resolution data and employ a noising-denoising process as introduced by *SDEdit* [28] on the samples from the base model. We follow [1] and specialize this refinement model on the first 200 (discrete) noise scales. During inference, we render latents from the base *SDXL*, and directly diffuse and denoise them in latent space with the refinement model (see Fig. 1), using the same text input. We note that this step is optional, but improves sample quality for detailed backgrounds and human faces, as demonstrated in Fig. 6 and Fig. 13.

To assess the performance of our model (with and without refinement stage), we conduct a user study, and let users pick their favorite generation from the following four models: *SDXL*, *SDXL* (with refiner), *Stable Diffusion* 1.5 and *Stable Diffusion* 2.1. The results demonstrate the *SDXL* with the refinement stage is the highest rated choice, and outperforms *Stable Diffusion* 1.5 & 2.1 by a significant margin (win rates: *SDXL* w/ refinement: 48.44%, *SDXL* base: 36.93%, *Stable Diffusion* 1.5: 7.91%, *Stable Diffusion* 2.1: 6.71%). See Fig. 1, which also provides an overview of the full pipeline. However, when using classical performance metrics such as FID and CLIP scores the improvements of *SDXL* over previous methods are not reflected as shown in Fig. 12 and discussed in App. F. This aligns with and further backs the findings of Kirstain et al. [23].



Figure 6:  $1024^2$  samples (with zoom-ins) from *SDXL* without (left) and with (right) the refinement model discussed. Prompt: “Epic long distance cityscape photo of New York City flooded by the ocean and overgrown buildings and jungle ruins in rainforest, at sunset, cinematic shot, highly detailed, 8k, golden light”. See Fig. 13 for additional samples.

### 3 Future Work

This report presents a preliminary analysis of improvements to the foundation model *Stable Diffusion* for text-to-image synthesis. While we achieve significant improvements in synthesized image quality, prompt adherence and composition, in the following, we discuss a few aspects for which we believe the model may be improved further:

- **Single stage:** Currently, we generate the best samples from *SDXL* using a two-stage approach with an additional refinement model. This results in having to load two large models into memory, hampering accessibility and sampling speed. Future work should investigate ways to provide a single stage of equal or better quality.
- **Text synthesis:** While the scale and the larger text encoder (OpenCLIP ViT-bigG [19]) help to improve the text rendering capabilities over previous versions of *Stable Diffusion*, incorporating byte-level tokenizers [52, 27] or simply scaling the model to larger sizes [53, 40] may further improve text synthesis.
- **Architecture:** During the exploration stage of this work, we briefly experimented with transformer-based architectures such as UViT [16] and DiT [33], but found no immediate benefit. We remain, however, optimistic that a careful hyperparameter study will eventually enable scaling to much larger transformer-dominated architectures.
- **Distillation:** While our improvements over the original *Stable Diffusion* model are significant, they come at the price of increased inference cost (both in VRAM and sampling speed). Future work will thus focus on decreasing the compute needed for inference, and increased sampling speed, for example through guidance- [29], knowledge- [6, 22, 24] and progressive distillation [41, 2, 29].
- Our model is trained in the discrete-time formulation of [14], and requires *offset-noise* [11, 25] for aesthetically pleasing results. The EDM-framework of Karras et al. [21] is a promising candidate for future model training, as its formulation in continuous time allows for increased sampling flexibility and does not require noise-schedule corrections.



# Appendix

## A Acknowledgements

We thank all the folks at StabilityAI who worked on comparisons, code, etc, in particular: Alex Goodwin, Benjamin Aubin, Bill Cusick, Dennis Nitrosocke Niedworok, Dominik Lorenz, Harry Saini, Ian Johnson, Ju Huo, Katie May, Mohamad Diab, Peter Baylies, Rahim Entezari, Yam Levi, Yannik Marek, Yizhou Zheng. We also thank ChatGPT for providing writing assistance.

## B Limitations

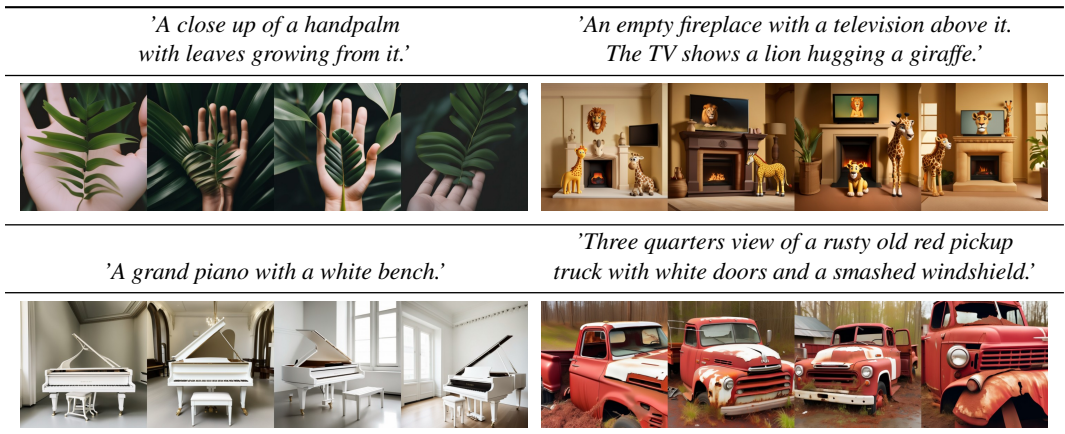


Figure 7: Failure cases of *SDXL* despite large improvements compared to previous versions of *Stable Diffusion*, the model sometimes still struggles with very complex prompts involving detailed spatial arrangements and detailed descriptions (e.g. top left example). Moreover, hands are not yet always correctly generated (e.g. top left) and the model sometimes suffers from two concepts bleeding into one another (e.g. bottom right example). All examples are random samples generated with 50 steps of the DDIM sampler [46] and cfg-scale 8.0 [13].

While our model has demonstrated impressive capabilities in generating realistic images and synthesizing complex scenes, it is important to acknowledge its inherent limitations. Understanding these limitations is crucial for further improvements and ensuring responsible use of the technology.

Firstly, the model may encounter challenges when synthesizing intricate structures, such as human hands (see Fig. 7, top left). Although it has been trained on a diverse range of data, the complexity of human anatomy poses a difficulty in achieving accurate representations consistently. This limitation suggests the need for further scaling and training techniques specifically targeting the synthesis of fine-grained details. A reason for this occurring might be that hands and similar objects appear with very high variance in photographs and it is hard for the model to extract the knowledge of the real 3D shape and physical limitations in that case.

Secondly, while the model achieves a remarkable level of realism in its generated images, it is important to note that it does not attain perfect photorealism. Certain nuances, such as subtle lighting effects or minute texture variations, may still be absent or less faithfully represented in the generated images. This limitation implies that caution should be exercised when relying solely on model-generated visuals for applications that require a high degree of visual fidelity.

Furthermore, the model’s training process heavily relies on large-scale datasets, which can inadvertently introduce social and racial biases. As a result, the model may inadvertently exacerbate these biases when generating images or inferring visual attributes.

In certain cases where samples contain multiple objects or subjects, the model may exhibit a phenomenon known as “concept bleeding”. This issue manifests as the unintended merging or overlap of distinct visual elements. For instance, in Fig. 14, an orange sunglass is observed, which indicates an instance of concept bleeding from the orange sweater. Another case of this can be seen in Fig. 8, the penguin is supposed to have a “blue hat” and “red gloves”, but is instead generated with blue

gloves and a red hat. Recognizing and addressing such occurrences is essential for refining the model’s ability to accurately separate and represent individual objects within complex scenes. The root cause of this may lie in the used pretrained text-encoders: firstly, they are trained to compress all information into a single token, so they may fail at binding only the right attributes and objects, Feng et al. [8] mitigate this issue by explicitly encoding word relationships into the encoding. Secondly, the contrastive loss may also contribute to this, since negative examples with a different binding are needed within the same batch [35].

Additionally, while our model represents a significant advancement over previous iterations of *SD*, it still encounters difficulties when rendering long, legible text. Occasionally, the generated text may contain random characters or exhibit inconsistencies, as illustrated in Fig. 8. Overcoming this limitation requires further investigation and development of techniques that enhance the model’s text generation capabilities, particularly for extended textual content — see for example the work of Liu et al. [27], who propose to enhance text rendering capabilities via character-level text tokenizers. Alternatively, scaling the model does further improve text synthesis [53, 40].

In conclusion, our model exhibits notable strengths in image synthesis, but it is not exempt from certain limitations. The challenges associated with synthesizing intricate structures, achieving perfect photorealism, further addressing biases, mitigating concept bleeding, and improving text rendering highlight avenues for future research and optimization.

## C Diffusion Models

In this section, we give a concise summary of DMs. We consider the continuous-time DM framework [47] and follow the presentation of Karras et al. [21]. Let  $p_{\text{data}}(\mathbf{x}_0)$  denote the data distribution and let  $p(\mathbf{x}; \sigma)$  be the distribution obtained by adding i.i.d.  $\sigma^2$ -variance Gaussian noise to the data. For sufficiently large  $\sigma_{\text{max}}$ ,  $p(\mathbf{x}; \sigma_{\text{max}}^2)$  is almost indistinguishable from  $\sigma_{\text{max}}^2$ -variance Gaussian noise. Capitalizing on this observation, DMs sample high variance Gaussian noise  $\mathbf{x}_M \sim \mathcal{N}(\mathbf{0}, \sigma_{\text{max}}^2)$  and sequentially denoise  $\mathbf{x}_M$  into  $\mathbf{x}_i \sim p(\mathbf{x}_i; \sigma_i)$ ,  $i \in \{0, \dots, M\}$ , with  $\sigma_i < \sigma_{i+1}$  and  $\sigma_M = \sigma_{\text{max}}$ . For a well-trained DM and  $\sigma_0 = 0$  the resulting  $\mathbf{x}_0$  is distributed according to the data.

**Sampling.** In practice, this iterative denoising process explained above can be implemented through the numerical simulation of the *Probability Flow* ordinary differential equation (ODE) [47]

$$d\mathbf{x} = -\dot{\sigma}(t)\sigma(t)\nabla_{\mathbf{x}} \log p(\mathbf{x}; \sigma(t)) dt, \quad (1)$$

where  $\nabla_{\mathbf{x}} \log p(\mathbf{x}; \sigma)$  is the *score function* [18]. The schedule  $\sigma(t): [0, 1] \rightarrow \mathbb{R}_+$  is user-specified and  $\dot{\sigma}(t)$  denotes the time derivative of  $\sigma(t)$ . Alternatively, we may also numerically simulate a stochastic differential equation (SDE) [47, 21]:

$$d\mathbf{x} = \underbrace{-\dot{\sigma}(t)\sigma(t)\nabla_{\mathbf{x}} \log p(\mathbf{x}; \sigma(t)) dt}_{\text{Probability Flow ODE; see Eq. (1)}} - \underbrace{\beta(t)\sigma^2(t)\nabla_{\mathbf{x}} \log p(\mathbf{x}; \sigma(t)) dt + \sqrt{2\beta(t)}\sigma(t) d\omega_t}_{\text{Langevin diffusion component}}, \quad (2)$$

where  $d\omega_t$  is the standard Wiener process. In principle, simulating either the Probability Flow ODE or the SDE above results in samples from the same distribution.

**Training.** DM training reduces to learning a model  $s_{\theta}(\mathbf{x}; \sigma)$  for the score function  $\nabla_{\mathbf{x}} \log p(\mathbf{x}; \sigma)$ . The model can, for example, be parameterized as  $\nabla_{\mathbf{x}} \log p(\mathbf{x}; \sigma) \approx s_{\theta}(\mathbf{x}; \sigma) = (D_{\theta}(\mathbf{x}; \sigma) - \mathbf{x})/\sigma^2$  [21], where  $D_{\theta}$  is a learnable *denoiser* that, given a noisy data point  $\mathbf{x}_0 + \mathbf{n}$ ,  $\mathbf{x}_0 \sim p_{\text{data}}(\mathbf{x}_0)$ ,  $\mathbf{n} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I}_d)$ , and conditioned on the noise level  $\sigma$ , tries to predict the clean  $\mathbf{x}_0$ . The denoiser  $D_{\theta}$  (or equivalently the score model) can be trained via *denoising score matching* (DSM)

$$\mathbb{E}_{(\mathbf{x}_0, \mathbf{c}) \sim p_{\text{data}}(\mathbf{x}_0, \mathbf{c}), (\sigma, \mathbf{n}) \sim p(\sigma, \mathbf{n})} [\lambda_{\sigma} \|D_{\theta}(\mathbf{x}_0 + \mathbf{n}; \sigma, \mathbf{c}) - \mathbf{x}_0\|_2^2], \quad (3)$$

where  $p(\sigma, \mathbf{n}) = p(\sigma) \mathcal{N}(\mathbf{n}; \mathbf{0}, \sigma^2)$ ,  $p(\sigma)$  is a distribution over noise levels  $\sigma$ ,  $\lambda_{\sigma}: \mathbb{R}_+ \rightarrow \mathbb{R}_+$  is a weighting function, and  $\mathbf{c}$  is an arbitrary conditioning signal, e.g., a class label, a text prompt, or a combination thereof. In this work, we choose  $p(\sigma)$  to be a discrete distributions over 1000 noise levels and set  $\lambda_{\sigma} = \sigma^{-2}$  similar to prior works [14, 38, 45].

**Classifier-free guidance.** Classifier-free guidance [13] is a technique to guide the iterative sampling process of a DM towards a conditioning signal  $\mathbf{c}$  by mixing the predictions of a conditional and an unconditional model

$$D^w(\mathbf{x}; \sigma, \mathbf{c}) = (1 + w)D(\mathbf{x}; \sigma, \mathbf{c}) - wD(\mathbf{x}; \sigma), \quad (4)$$

where  $w \geq 0$  is the *guidance strength*. In practice, the unconditional model can be trained jointly alongside the conditional model in a single network by randomly replacing the conditional signal  $\mathbf{c}$  with a null embedding in Eq. (3), e.g., 10% of the time [13]. Classifier-free guidance is widely used to improve the sampling quality, trading for diversity, of text-to-image DMs [30, 38].

## D Comparison to the State of the Art

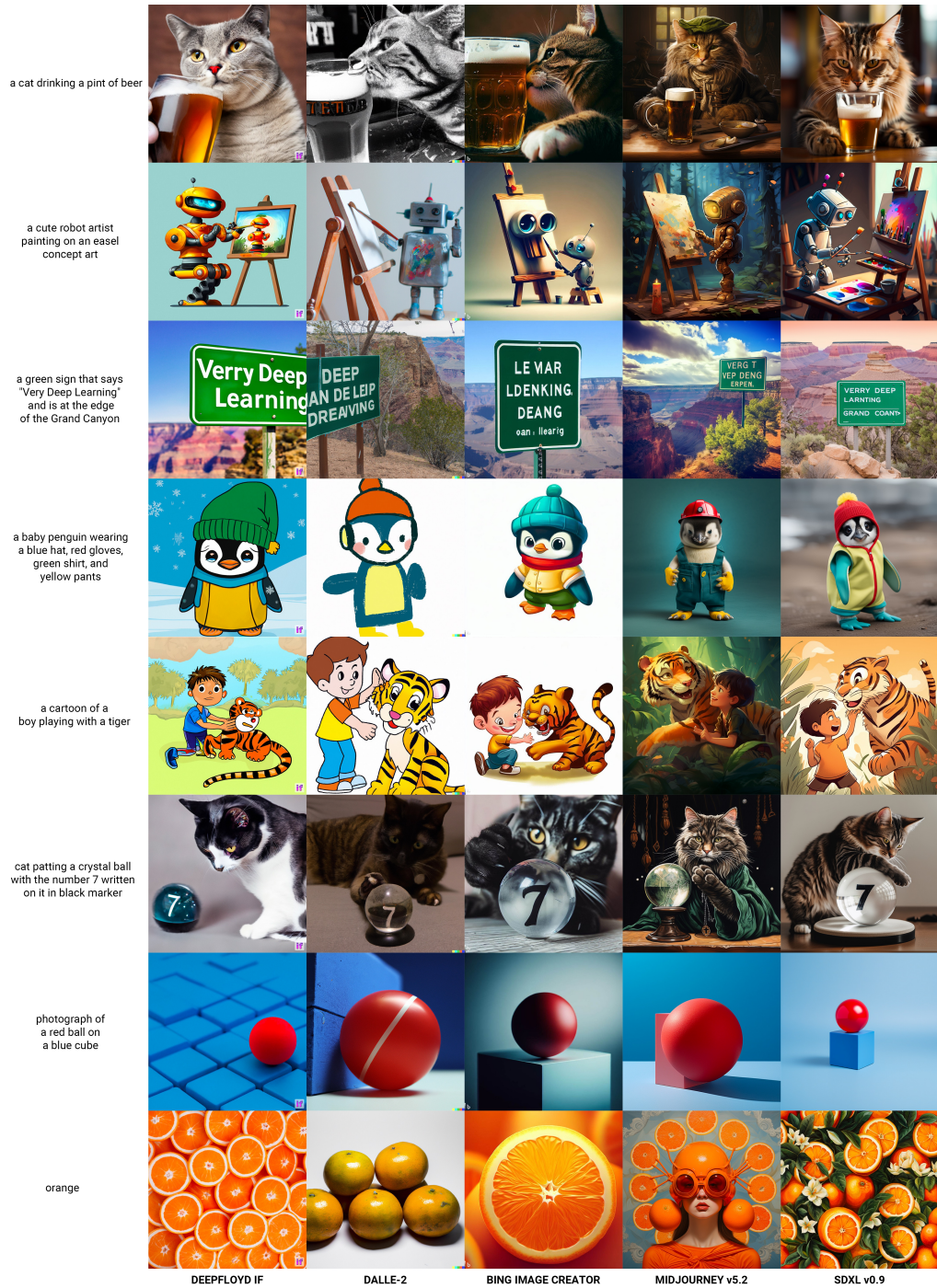


Figure 8: Qualitative comparison of *SDXL* with DeepFloyd IF, DALLE-2, Bing Image Creator, and Midjourney v5.2. To mitigate any bias arising from cherry-picking, Parti (P2) prompts were randomly selected. Seed 3 was uniformly applied across all models in which such a parameter could be designated. For models without a seed-setting feature, the first generated image is included.

## E Comparison to Midjourney v5.1

### E.1 Overall Votes

To assess the generation quality of *SDXL* we perform a user study against the state of the art text-to-image generation platform Midjourney<sup>1</sup>. As the source for image captions we use the PartiPrompts (P2) benchmark [53], that was introduced to compare large text-to-image model on various challenging prompts.

For our study, we choose five random prompts from each category, and generate four  $1024 \times 1024$  images by both Midjourney (v5.1, with a set seed of 2) and *SDXL* for each prompt. These images were then presented to the AWS GroundTruth taskforce, who voted based on adherence to the prompt. The results of these votes are illustrated in Fig. 9. Overall, there is a slight preference for *SDXL* over Midjourney in terms of prompt adherence.

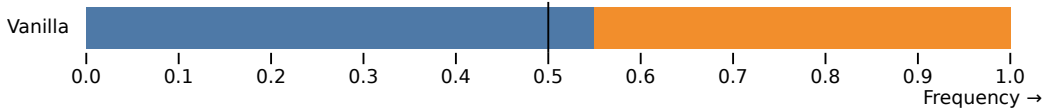


Figure 9: Results from 17,153 user preference comparisons between *SDXL* v0.9 and *Midjourney* v5.1, which was the latest version available at the time. The comparisons span all “categories” and “challenges” in the PartiPrompts (P2) benchmark. Notably, *SDXL* was favored 54.9% of the time over *Midjourney* V5.1. Preliminary testing indicates that the recently-released *Midjourney* V5.2 has lower prompt comprehension than its predecessor, but the laborious process of generating multiple prompts hampers the speed of conducting broader tests.

### E.2 Category & challenge comparisons on PartiPrompts (P2)

Each prompt from the P2 benchmark is organized into a category and a challenge, each focus on different difficult aspects of the generation process. We show the comparisons for each category (Fig. 10) and challenge (Fig. 11) of P2 below. In four out of six categories *SDXL* outperforms Midjourney, and in seven out of ten challenges there is no significant difference between both models or *SDXL* outperforms Midjourney.

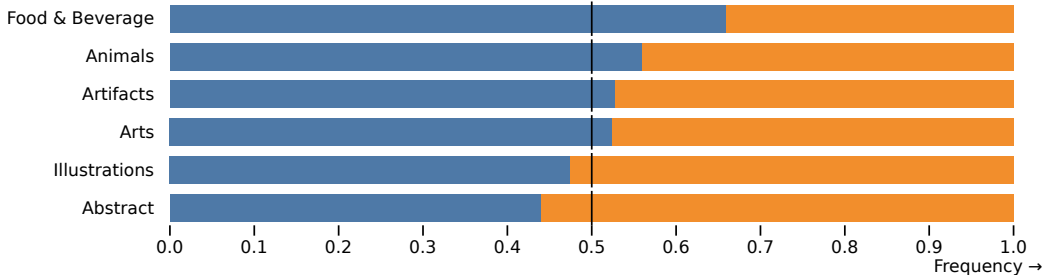


Figure 10: User preference comparison of *SDXL* (without refinement model) and *Midjourney* V5.1 across particular text categories. *SDXL* outperforms *Midjourney* V5.1 in all but two categories.

<sup>1</sup>We compare against v5.1 since that was the best version available at that time.

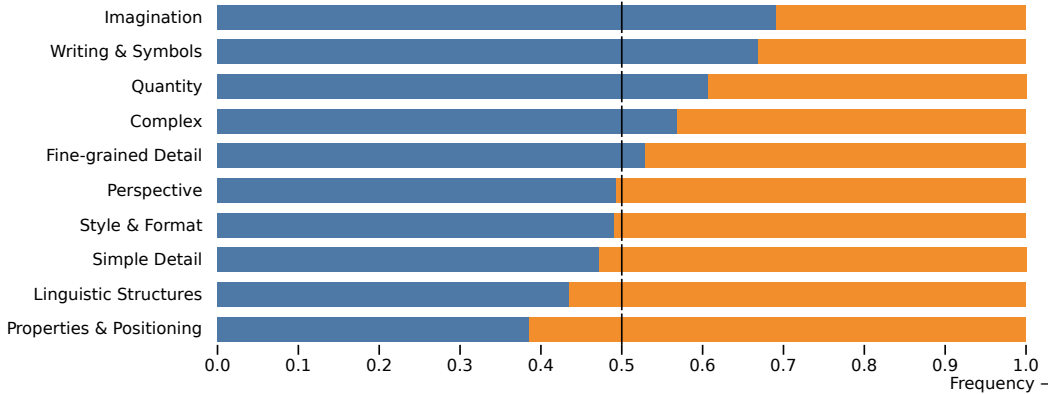


Figure 11: Preference comparisons of *SDXL* (with refinement model) to *Midjourney V5.1* on complex prompts. *SDXL* either outperforms or is statistically equal to *Midjourney V5.1* in 7 out of 10 categories.

## F On FID Assessment of Generative Text-Image Foundation Models

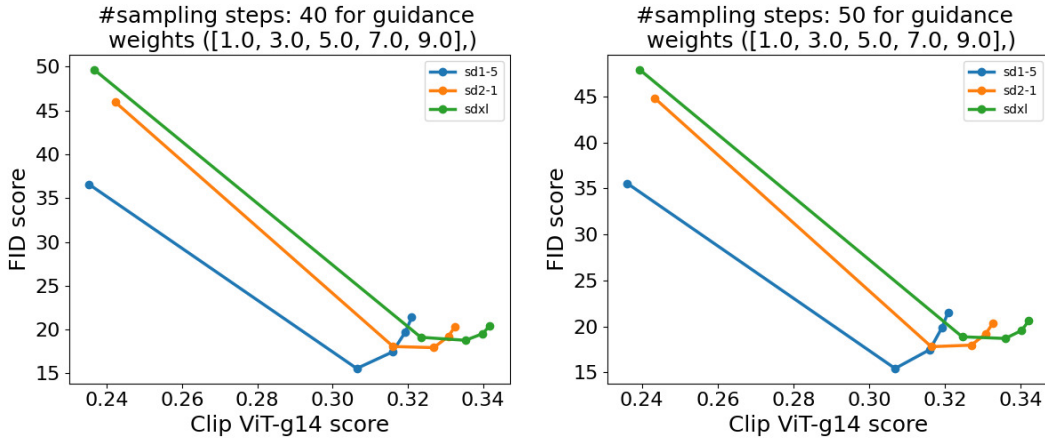


Figure 12: Plotting FID vs CLIP score for different cfg scales. *SDXL* shows only slightly improved text-alignment, as measured by CLIP-score, compared to previous versions that do not align with the judgement of human evaluators. Even further and similar as in [23], FID are worse than for both *SD-1.5* and *SD-2.1*, while human evaluators clearly prefer the generations of *SD-XL* over those of these previous models.

Throughout the last years it has been common practice for generative text-to-image models to assess FID- [12] and CLIP-scores [34, 36] in a zero-shot setting on complex, small-scale text-image datasets of natural images such as COCO [26]. However, with the advent of foundational text-to-image models [40, 37, 38, 1], which are not only targeting visual compositionality, but also at other difficult tasks such as deep text understanding, fine-grained distinction between unique artistic styles and especially a pronounced sense of visual aesthetics, this particular form of model evaluation has become more and more questionable. Kirstain et al. [23] demonstrates that COCO zero-shot FID is *negatively correlated* with visual aesthetics, and such measuring the generative performance of such models should be rather done by human evaluators. We investigate this for *SDXL* and visualize FID-vs-CLIP curves in Fig. 12 for 10k text-image pairs from COCO [26]. Despite its drastically improved performance as measured quantitatively by asking human assessors (see Fig. 1) as well as qualitatively (see Fig. 4 and Fig. 14), *SDXL* does *not* achieve better FID scores than the previous *SD* versions. Contrarily, FID for *SDXL* is the worst of all three compared models while only showing slightly improved CLIP-scores (measured with OpenClip ViT g-14). Thus, our results back the findings of Kirstain et al. [23] and further emphasize the need for additional quantitative performance scores, specifically for text-to-image foundation models. All scores have been evaluated based on 10k generated examples.

## G Additional Comparison between Single- and Two-Stage *SDXL* pipeline

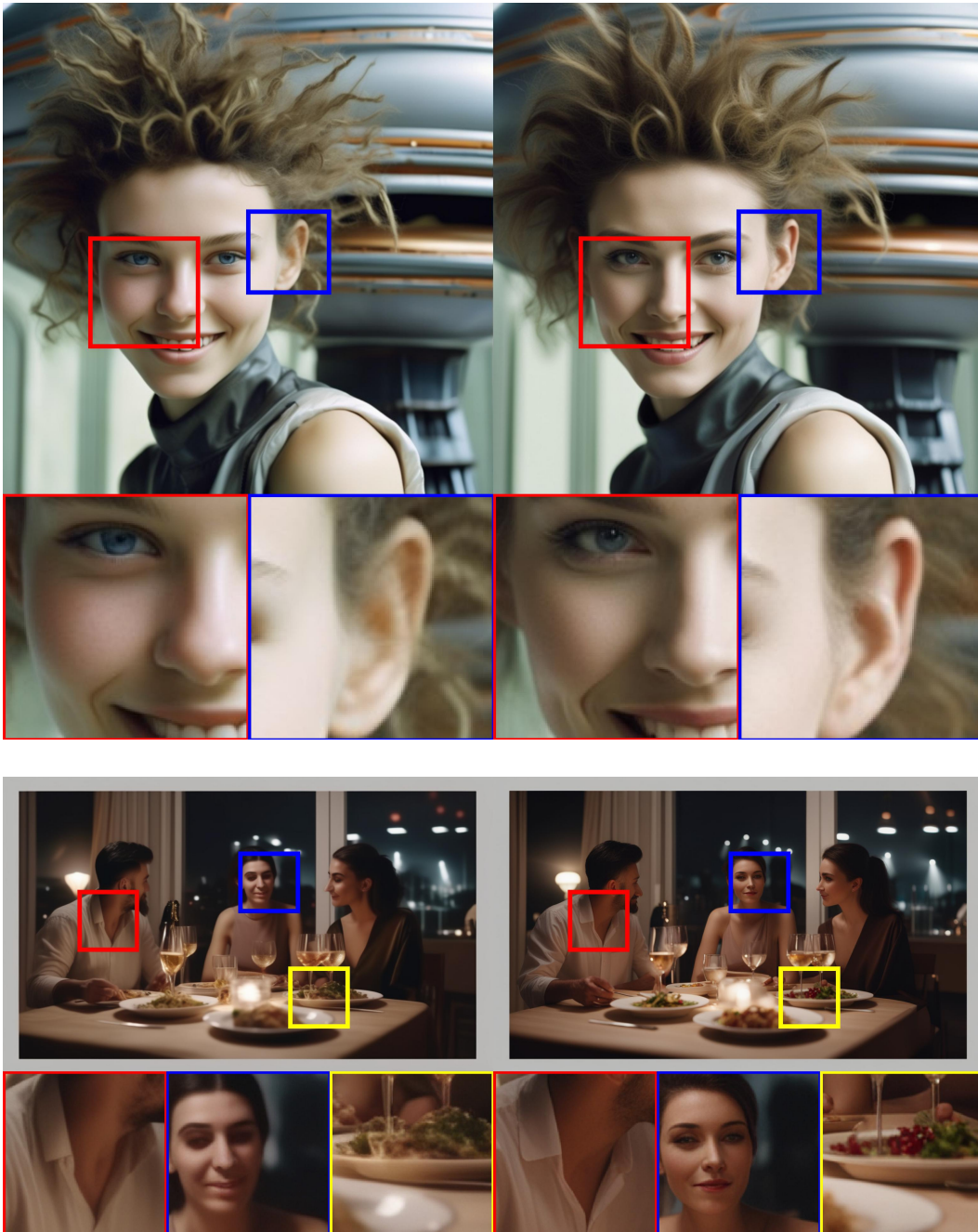


Figure 13: *SDXL* samples (with zoom-ins) without (left) and with (right) the refinement model discussed. Prompt: (top) “close up headshot, futuristic young woman, wild hair sly smile in front of gigantic UFO, dslr, sharp focus, dynamic composition” (bottom) “Three people having dinner at a table at new years eve, cinematic shot, 8k”. Zoom-in for details.

## H Comparison between *SD 1.5* vs. *SD 2.1* vs. *SDXL*

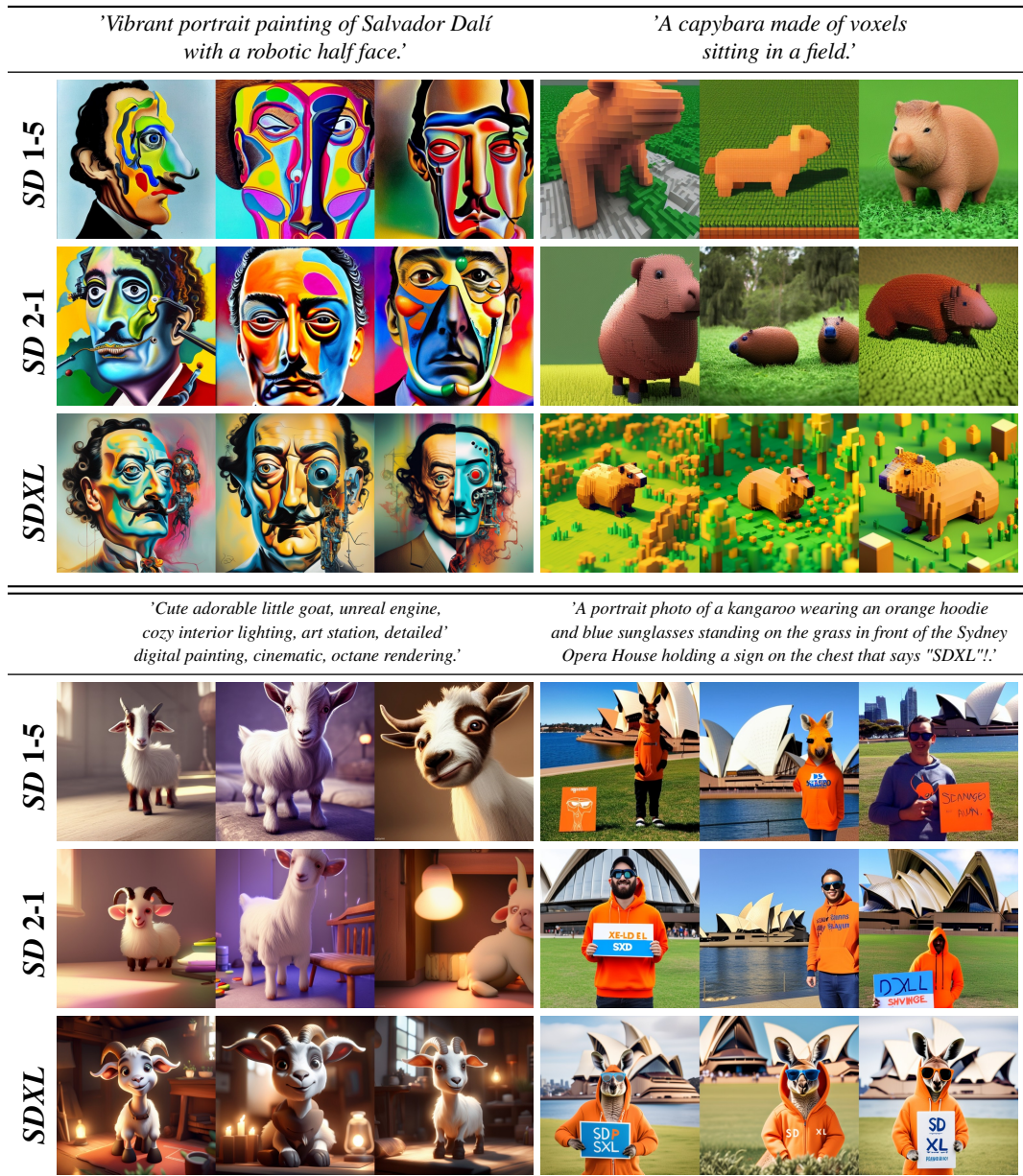


Figure 14: Additional results for the comparison of the output of *SDXL* with previous versions of *Stable Diffusion*. For each prompt, we show 3 random samples of the respective model for 50 steps of the DDIM sampler [46] and cfg-scale 8.0 [13]





Figure 15: Additional results for the comparison of the output of *SDXL* with previous versions of *Stable Diffusion*. For each prompt, we show 3 random samples of the respective model for 50 steps of the DDIM sampler [46] and cfg-scale 8.0 [13].

## I Multi-Aspect Training Hyperparameters

We use the following image resolutions for mixed-aspect ratio finetuning as described in Sec. 2.3.

Height	Width	Aspect Ratio	Height	Width	Aspect Ratio
512	2048	0.25	1024	1024	1.0
512	1984	0.26	1024	960	1.07
512	1920	0.27	1088	960	1.13
512	1856	0.28	1088	896	1.21
576	1792	0.32	1152	896	1.29
576	1728	0.33	1152	832	1.38
576	1664	0.35	1216	832	1.46
640	1600	0.4	1280	768	1.67
640	1536	0.42	1344	768	1.75
704	1472	0.48	1408	704	2.0
704	1408	0.5	1472	704	2.09
704	1344	0.52	1536	640	2.4
768	1344	0.57	1600	640	2.5
768	1280	0.6	1664	576	2.89
832	1216	0.68	1728	576	3.0
832	1152	0.72	1792	576	3.11
896	1152	0.78	1856	512	3.62
896	1088	0.82	1920	512	3.75
960	1088	0.88	1984	512	3.88
960	1024	0.94	2048	512	4.0

## J Pseudo-code for Conditioning Concatenation along the Channel Axis

```
1 from einops import rearrange
2 import torch
3
4 batch_size=16
5 # channel dimension of pooled output of text encoder(s)
6 pooled_dim = 512
7
8 def fourier_embedding(inputs, outdim=256, max_period=10000):
9     """
10    Classical sinusoidal timestep embedding
11    as commonly used in diffusion models
12    :param inputs: batch of integer scalars shape [b,]
13    :param outdim: embedding dimension
14    :param max_period: max freq added
15    :return: batch of embeddings of shape [b, outdim]
16    """
17    ...
18
19 def cat_along_channel_dim(
20     x:torch.Tensor,) -> torch.Tensor:
21     if x.ndim == 1:
22         x = x[... ,None]
23     assert x.ndim == 2
24     b, d_in = x.shape
25     x = rearrange(x, "b din -> (b din)")
26     # fourier fn adds additional dimension
27     emb = fourier_embedding(x)
28     d_f = emb.shape[-1]
29     emb = rearrange(emb, "(b din) df -> b (din df)",
30                    b=b, din=d_in, df=d_f)
31     return emb
32
33 def concat_embeddings(
34     # batch of size and crop conditioning cf. Sec. 3.2
35     c_size:torch.Tensor,
36     c_crop:torch.Tensor,
37     # batch of aspect ratio conditioning cf. Sec. 3.3
38     c_ar:torch.Tensor,
39     # final output of text encoders after pooling cf. Sec. 3.1
40     c_pooled_txt:torch.Tensor, ) -> torch.Tensor:
41     # fourier feature for size conditioning
42     c_size_emb = cat_along_channel_dim(c_size)
43     # fourier feature for size conditioning
44     c_crop_emb = cat_along_channel_dim(c_crop)
45     # fourier feature for size conditioning
46     c_ar_emb = cat_along_channel_dim(c_ar)
47     # the concatenated output is mapped to the same
48     # channel dimension than the noise level conditioning
49     # and added to that conditioning before being fed to the unet
50     return torch.cat([c_pooled_txt,
51                     c_size_emb,
52                     c_crop_emb,
53                     c_ar_emb], dim=1)
54
55 # simulating c_size and c_crop as in Sec. 3.2
56 c_size=torch.zeros((batch_size, 2)).long()
57 c_crop=torch.zeros((batch_size, 2)).long()
58 # simulating c_ar and pooled text encoder output as in Sec. 3.3
59 c_ar=torch.zeros((batch_size, 2)).long()
60 c_pooled=torch.zeros((batch_size, pooled_dim)).long()
61
62 # get concatenated embedding
63 c_concat = concat_embeddings(c_size, c_crop, c_ar, c_pooled)
```

Figure 16: Python code for concatenating the additional conditionings introduced in Secs. 2.1 to 2.3 along the channel dimension.

## References

- [1] Yogesh Balaji, Seungjun Nah, Xun Huang, Arash Vahdat, Jiaming Song, Karsten Kreis, Miika Aittala, Timo Aila, Samuli Laine, Bryan Catanzaro, Tero Karras, and Ming-Yu Liu. eDiff-I: Text-to-Image Diffusion Models with an Ensemble of Expert Denoisers. *arXiv:2211.01324*, 2022.
- [2] David Berthelot, Arnaud Autef, Jierui Lin, Dian Ang Yap, Shuangfei Zhai, Siyuan Hu, Daniel Zheng, Walter Talbot, and Eric Gu. TRACT: Denoising Diffusion Models with Transitive Closure Time-Distillation. *arXiv:2303.04248*, 2023.
- [3] Andreas Blattmann, Robin Rombach, Huan Ling, Tim Dockhorn, Seung Wook Kim, Sanja Fidler, and Karsten Kreis. Align your Latents: High-Resolution Video Synthesis with Latent Diffusion Models. *arXiv:2304.08818*, 2023.
- [4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.
- [5] Prafulla Dhariwal and Alex Nichol. Diffusion Models Beat GANs on Image Synthesis. *arXiv:2105.05233*, 2021.
- [6] Tim Dockhorn, Robin Rombach, Andreas Blattmann, and Yaoliang Yu. Distilling the Knowledge in Diffusion Models. *CVPR Workshop on Generative Models for Computer Vision*, 2023.
- [7] Patrick Esser, Johnathan Chiu, Parmida Atighehchian, Jonathan Granskog, and Anastasis Germanidis. Structure and content-guided video synthesis with diffusion models, 2023.
- [8] Weixi Feng, Xuehai He, Tsu-Jui Fu, Varun Jampani, Arjun Akula, Pradyumna Narayana, Sugato Basu, Xin Eric Wang, and William Yang Wang. Training-free structured diffusion guidance for compositional text-to-image synthesis. *arXiv:2212.05032*, 2023.
- [9] Seth Forsgren and Hayk Martiros. Riffusion - Stable diffusion for real-time music generation, 2022. URL <https://riffusion.com/about>.
- [10] 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.
- [11] Nicholas Guttenberg and CrossLabs. Diffusion with offset noise, 2023. URL <https://www.crosslabs.org/blog/diffusion-with-offset-noise>.
- [12] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. *arXiv:1706.08500*, 2017.
- [13] Jonathan Ho and Tim Salimans. Classifier-Free Diffusion Guidance. *arXiv:2207.12598*, 2022.
- [14] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising Diffusion Probabilistic Models. *arXiv preprint arXiv:2006.11239*, 2020.
- [15] Jonathan Ho, William Chan, Chitwan Saharia, Jay Whang, Ruiqi Gao, Alexey Gritsenko, Diederik P Kingma, Ben Poole, Mohammad Norouzi, David J Fleet, and Tim Salimans. Imagen Video: High Definition Video Generation with Diffusion Models. *arXiv:2210.02303*, 2022.
- [16] Emiel Hoogeboom, Jonathan Heek, and Tim Salimans. simple diffusion: End-to-end diffusion for high resolution images. *arXiv preprint arXiv:2301.11093*, 2023.
- [17] Rongjie Huang, Jiawei Huang, Dongchao Yang, Yi Ren, Luping Liu, Mingze Li, Zhenhui Ye, Jinglin Liu, Xiang Yin, and Zhou Zhao. Make-An-Audio: Text-To-Audio Generation with Prompt-Enhanced Diffusion Models. *arXiv:2301.12661*, 2023.
- [18] Aapo Hyvärinen and Peter Dayan. Estimation of Non-Normalized Statistical Models by Score Matching. *Journal of Machine Learning Research*, 6(4), 2005.
- [19] Gabriel Ilharco, Mitchell Wortsman, Ross Wightman, Cade Gordon, Nicholas Carlini, Rohan Taori, Achal Dave, Vaishaal Shankar, Hongseok Namkoong, John Miller, Hannaneh Hajishirzi, Ali Farhadi, and Ludwig Schmidt. OpenCLIP, July 2021. URL <https://doi.org/10.5281/zenodo.5143773>.
- [20] Heewoo Jun, Rewon Child, Mark Chen, John Schulman, Aditya Ramesh, Alec Radford, and Ilya Sutskever. Distribution Augmentation for Generative Modeling. In *International Conference on Machine Learning*, pages 5006–5019. PMLR, 2020.
- [21] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the Design Space of Diffusion-Based Generative Models. *arXiv:2206.00364*, 2022.
- [22] Bo-Kyeong Kim, Hyoung-Kyu Song, Thibault Castells, and Shinkook Choi. On Architectural Compression of Text-to-Image Diffusion Models. *arXiv:2305.15798*, 2023.

- [23] Yuval Kirstain, Adam Polyak, Uriel Singer, Shahbuland Matiana, Joe Penna, and Omer Levy. Pick-a-pic: An open dataset of user preferences for text-to-image generation. *arXiv:2305.01569*, 2023.
- [24] Yanyu Li, Huan Wang, Qing Jin, Ju Hu, Pavlo Chemerys, Yun Fu, Yanzhi Wang, Sergey Tulyakov, and Jian Ren. SnapFusion: Text-to-Image Diffusion Model on Mobile Devices within Two Seconds. *arXiv:2306.00980*, 2023.
- [25] Shanchuan Lin, Bingchen Liu, Jiashi Li, and Xiao Yang. Common Diffusion Noise Schedules and Sample Steps are Flawed. *arXiv:2305.08891*, 2023.
- [26] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2015.
- [27] Rosanne Liu, Dan Garrette, Chitwan Saharia, William Chan, Adam Roberts, Sharan Narang, Irina Blok, RJ Mical, Mohammad Norouzi, and Noah Constant. Character-aware models improve visual text rendering, 2023.
- [28] Chenlin Meng, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. SDEdit: Guided Image Synthesis and Editing with Stochastic Differential Equations. *arXiv:2108.01073*, 2021.
- [29] Chenlin Meng, Robin Rombach, Ruiqi Gao, Diederik P. Kingma, Stefano Ermon, Jonathan Ho, and Tim Salimans. On distillation of guided diffusion models, 2023.
- [30] Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models. *arXiv:2112.10741*, 2021.
- [31] NovelAI. Novelai improvements on stable diffusion, 2023. URL <https://blog.novelai.net/novelai-improvements-on-stable-diffusion-e10d38db82ac>.
- [32] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019.
- [33] William Peebles and Saining Xie. Scalable Diffusion Models with Transformers. *arXiv:2212.09748*, 2022.
- [34] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning Transferable Visual Models From Natural Language Supervision. *arXiv:2103.00020*, 2021.
- [35] Aditya Ramesh. How dall-e 2 works, 2022. URL <http://adityaramesh.com/posts/dalle2/dalle2.html>.
- [36] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation, 2021.
- [37] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical Text-Conditional Image Generation with CLIP Latents. *arXiv:2204.06125*, 2022.
- [38] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-Resolution Image Synthesis with Latent Diffusion Models. *arXiv preprint arXiv:2112.10752*, 2021.
- [39] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. *arXiv:1505.04597*, 2015.
- [40] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S Sara Mahdavi, Rapha Gontijo Lopes, Tim Salimans, Jonathan Ho, David J Fleet, and Mohammad Norouzi. Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding. *arXiv:2205.11487*, 2022.
- [41] Tim Salimans and Jonathan Ho. Progressive Distillation for Fast Sampling of Diffusion Models. *arXiv preprint arXiv:2202.00512*, 2022.
- [42] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved Techniques for Training GANs. *arXiv:1606.03498*, 2016.
- [43] Sitian Shen, Zilin Zhu, Linqian Fan, Harry Zhang, and Xinxiao Wu. DiffCLIP: Leveraging Stable Diffusion for Language Grounded 3D Classification. *arXiv:2305.15957*, 2023.
- [44] Uriel Singer, Adam Polyak, Thomas Hayes, Xi Yin, Jie An, Songyang Zhang, Qiyuan Hu, Harry Yang, Oron Ashual, Oran Gafni, Devi Parikh, Sonal Gupta, and Yaniv Taigman. Make-A-Video: Text-to-Video Generation without Text-Video Data. *arXiv:2209.14792*, 2022.
- [45] Jascha Sohl-Dickstein, Eric A Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep Unsupervised Learning using Nonequilibrium Thermodynamics. *arXiv:1503.03585*, 2015.

- [46] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv:2010.02502*, 2020.
- [47] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-Based Generative Modeling through Stochastic Differential Equations. *arXiv:2011.13456*, 2020.
- [48] Andreas Stöckl. Evaluating a synthetic image dataset generated with stable diffusion. *arXiv:2211.01777*, 2022.
- [49] Yu Takagi and Shinji Nishimoto. High-Resolution Image Reconstruction With Latent Diffusion Models From Human Brain Activity. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14453–14463, 2023.
- [50] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. LLaMA: Open and Efficient Foundation Language Models. *arXiv:2302.13971*, 2023.
- [51] Jialiang Wei, Anne-Lise Courbis, Thomas Lambolais, Binbin Xu, Pierre Louis Bernard, and Gérard Dray. Boosting gui prototyping with diffusion models. *arXiv preprint arXiv:2306.06233*, 2023.
- [52] Linting Xue, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, and Colin Raffel. Byt5: Towards a token-free future with pre-trained byte-to-byte models, 2022.
- [53] Jiahui Yu, Yuanzhong Xu, Jing Yu Koh, Thang Luong, Gunjan Baid, Zirui Wang, Vijay Vasudevan, Alexander Ku, Yinfei Yang, Burcu Karagol Ayan, Ben Hutchinson, Wei Han, Zarana Parekh, Xin Li, Han Zhang, Jason Baldridge, and Yonghui Wu. Scaling autoregressive models for content-rich text-to-image generation, 2022.
- [54] Lvmin Zhang and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. *arXiv:2302.05543*, 2023.
- [55] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric, 2018.