Hierarchical Text-Conditional Image Generation with CLIP Latents

Aditya Ramesh∗ Prafulla Dhariwal∗ Alex Nichol∗
OpenAI OpenAI OpenAI
aramesh@openai.com prafulla@openai.com alex@openai.com

Casey Chu∗
OpenAI
casey@openai.com

Mark Chen
OpenAI
mark@openai.com

Abstract

Contrastive models like CLIP have been shown to learn robust representations of images that capture both semantics and style. To leverage these representations for image generation, we propose a two-stage model: a prior that generates a CLIP image embedding given a text caption, and a decoder that generates an image conditioned on the image embedding. We show that explicitly generating image representations improves image diversity with minimal loss in photorealism and caption similarity. Our decoders conditioned on image representations can also produce variations of an image that preserve both its semantics and style, while varying the non-essential details absent from the image representation. We use diffusion models for the decoder and experiment with both autoregressive and diffusion models for the prior, finding that the latter are computationally more efficient and produce higher-quality samples.

1 Introduction

Recent progress in computer vision has been driven by scaling models on large datasets of captioned images collected from the internet [10, 44, 60, 39, 31, 16]. Within this framework, CLIP [39] has emerged as a successful representation learner for images. CLIP embeddings have a number of desirable properties: they are robust to image distribution shift, have impressive zero-shot capabilities, and have been fine-tuned to achieve state-of-the-art results on a wide variety of vision and language tasks [45]. Concurrently, diffusion models [46, 48, 25] have emerged as a promising generative modeling framework, pushing the state-of-the-art on image and video generation tasks [11, 26, 24]. To achieve best results, diffusion models leverage a guidance technique [11, 24] which improves sample fidelity (for images, photorealism) at the cost of sample diversity.

In this work, we combine these two approaches for the problem of text-conditional image generation. We first train a diffusion decoder to invert the CLIP image encoder. Our inverter is non-deterministic, and can produce multiple images corresponding to a given image embedding. The presence of an encoder and its approximate inverse (the decoder) allows for capabilities beyond text-to-image translation. As in GAN inversion [62, 55], encoding and decoding an input image produces semantically similar output images. We can also interpolate between input images by inverting interpolations of their image embeddings. However, one notable advantage of using the CLIP latent space is the ability to semantically modify images by moving in the direction of any encoded text vector, whereas discovering these directions in GAN latent space involves luck and diligent manual examination.

∗Equal contribution
Figure 1: Selected 1024 × 1024 samples from a production version of our model.
Figure 2: Variations of an input image by encoding with CLIP and then decoding with a diffusion model. The variations preserve both semantic information like presence of a clock in the painting and the overlapping strokes in the logo, as well as stylistic elements like the surrealism in the painting and the color gradients in the logo, while varying the non-essential details.

Furthermore, encoding and decoding images also provides us with a tool for observing which features of the image are recognized or disregarded by CLIP.

To obtain a full generative model of images, we combine the CLIP image embedding decoder with a prior model, which generates possible CLIP image embeddings from a given text caption. We compare our text-to-image system with other systems such as DALL-E [40] and GLIDE [35], finding that our samples are comparable in quality to GLIDE, but with greater diversity in our generations. We also develop methods for training diffusion priors in latent space, and show that they achieve comparable performance to autoregressive priors, while training much more efficiently. We refer to our full text-conditional image generation stack as unCLIP, since it generates images by inverting the CLIP image encoder.

2 Method

Our training dataset consists of pairs \((x, y)\) of images \(x\) and their corresponding captions \(y\). Given an image \(x\), let \(z_i\) and \(z_t\) be its CLIP image and text embeddings, respectively. We design our generative stack to produce images from captions using two components:

- A prior \(P(z_i|y)\) that produces CLIP image embeddings \(z_i\) conditioned on captions \(y\).
Figure 3: A high-level overview of unCLIP. Above the dotted line, we depict the CLIP training process, through which we learn a joint representation space for text and images. Below the dotted line, we depict our text-to-image generation process: a CLIP text embedding is first fed to an autoregressive or diffusion prior to produce an image embedding, and then this embedding is used to condition a diffusion decoder which produces a final image. Note that the CLIP model is frozen during training of the prior and decoder.

- **A decoder** $P(x|z_i, y)$ that produces images $x$ conditioned on CLIP image embeddings $z_i$ (and optionally text captions $y$).

The decoder allows us to invert images given their CLIP image embeddings, while the prior allows us to learn a generative model of the image embeddings themselves. Stacking these two components yields a generative model $P(x|y)$ of images $x$ given captions $y$:

$$P(x|y) = P(x, z_i|y) = P(x|z_i, y)P(z_i|y).$$

The first equality holds because $z_i$ is a deterministic function of $x$. The second equality holds because of the chain rule. Thus, we can sample from the true conditional distribution $P(x|y)$ by first sampling $z_i$ using the prior, and then sampling $x$ using the decoder. In the following sections, we describe our decoder and prior stacks. For training details and hyperparameters, refer to Appendix C.

### 2.1 Decoder

We use diffusion models [25, 48] to produce images conditioned on CLIP image embeddings (and optionally text captions). Specifically, we modify the architecture described in Nichol et al. (2021) by projecting and adding CLIP embeddings to the existing timestep embedding, and by projecting CLIP embeddings into four extra tokens of context that are concatenated to the sequence of outputs from the GLIDE text encoder. We retained the text conditioning pathway present in the original GLIDE model, hypothesizing that it could allow the diffusion model to learn aspects of natural language that CLIP fails to capture (e.g. variable binding), but find that it offers little help in this regard (Section 6). We enable classifier-free guidance [24] by randomly setting the CLIP embeddings to zero (or a learned embedding) 10% of the time, and randomly dropping the text caption 50% of the time.

To generate high resolution images, we train two diffusion upsampler models [34, 43]: one to upsample images from $64 \times 64$ to $256 \times 256$, and another to further upsample those to $1024 \times 1024$. To improve the robustness of our upsamplers, we slightly corrupt the conditioning images during training. For the first upsampling stage, we use gaussian blur [43], and for the second, we use a more diverse BSR degradation [42].
To reduce training compute and improve numerical stability, we follow Rombach et al. [42] and train on random crops of images that are one-fourth the target size. We use only spatial convolutions in the model and at inference time directly apply the model at the target resolution, observing that it readily generalizes to the higher resolution. We found no benefit from conditioning the upsamplers on the caption, and use unconditional ADMNets [11].

2.2 Prior

While a decoder can invert CLIP image embeddings $z_i$ to produce images $x$, we need a prior model that produces $z_i$ from captions $y$ to enable image generations from text captions. We explore two different model classes for the prior model:

- **Autoregressive (AR) prior:** the CLIP image embedding $z_i$ is converted into a sequence of discrete codes and predicted autoregressively conditioned on the caption $y$.
- **Diffusion prior:** The continuous vector $z_i$ is directly modelled using a Gaussian diffusion model conditioned on the caption $y$.

In addition to the caption, we can condition on the CLIP text embedding $z_t$ since it is a deterministic function of the caption. We additionally allow for sampling using classifier-free guidance for both the AR and diffusion prior by randomly dropping this text conditioning information during training.

To train and sample from the AR prior more efficiently, we first reduce the dimensionality of the CLIP image embeddings $z_i$ by applying Principal Component Analysis (PCA) [37]. In particular, we find that the rank of the CLIP representation space is drastically reduced when training CLIP with SAM [15] while slightly improving evaluation metrics. We are able to preserve nearly all of the information by retaining only 319 principal components out of the original 1,024. After applying PCA, we order the principal components by decreasing eigenvalue magnitude, quantize each of the 319 dimensions into 1,024 discrete buckets, and predict the resulting sequence using a decoder-only Transformer [53] model with a causal attention mask.

For the diffusion prior, we train a decoder-only Transformer with a causal attention mask on a sequence consisting of encoded text, the CLIP text embedding, an embedding for the diffusion timestep, the noised CLIP image embedding, and a final embedding whose output from the Transformer is used to predict the unnoised CLIP image embedding. We choose not to condition the diffusion prior on $z_i \cdot z_t$, like in the AR prior; instead, we improve quality during sampling time by generating two samples of $z_i$ and selecting the one with a higher dot product with $z_t$. Instead of using the $\epsilon$-prediction formulation from Ho et al. [25], we find it better to train our model to predict the unnoised $z_i$ directly, and use a mean-squared error loss on this prediction:

$$L_{\text{prior}} = \mathbb{E}_{t \sim [1,T], z_i^{(t)} \sim q_t} \left[ \| f_\theta(z_i^{(t)}, t, y) - z_i \|^2 \right]$$

2.3 Variations

Given an image $x$, we can obtain its CLIP image embedding $z_i$ and then use our decoder to “invert” $z_i$, producing new images that we call variations of our input. To make the variations more similar to the original

---

\(^2\)I.e., less than 1% average mean-squared error in reconstructing the image representations.

\(^3\)We swept over percentiles 50%, 70%, 85%, 95% and found 50% to be optimal in all experiments.
image $x$, we first perform DDIM inversion \cite{ho2020denoising} of $x$ with the decoder conditioned on $z_i$ to obtain its DDIM inverted latent $z_T$. We can then use $z_T$ as the initial noise when generating the images through the decoder. We use DDIM \cite{ho2020denoising} with $\eta > 0$ for sampling, with higher values of $\eta$ giving rise to more diverse variations. It is also possible to combine two images for variations. To do so, we perform spherical interpolation of their CLIP embeddings $z_i$ and $z_j$ to obtain intermediate $z_\theta = \text{slerp}(z_i, z_j, \theta)$, and produce variations of $z_\theta$ by passing it through the decoder.

3 Exploring the CLIP Latent Space Using the unCLIP Decoder

Our decoder model provides a unique opportunity to explore CLIP latent space by allowing us to directly visualize what the CLIP image encoder is seeing. In Figure 2, we encode an image using CLIP, and then decode its image embedding using the diffusion decoder, obtaining variations of the image (Section 2.3). These variations tell us what information was captured in the CLIP image embedding (and thus is preserved across samples), and what was lost (and thus changes across the samples). One can also interpolate between CLIP embeddings to mix information between two images, as shown in Figure 4.

As an example use case, we can revisit cases where CLIP makes incorrect predictions, such as typographic attacks \cite{khan2022typo}. In these adversarial images, a piece of text is overlayed on top of an object, which causes CLIP to predict the object described by the text rather than the object depicted in the image. This piece of text essentially hides the original object in terms of output probabilities. In Figure 5, we show an example of this attack from \cite{khan2022typo}, wherein an apple can be misclassified as an iPod. Surprisingly, we find that our decoder still generates pictures of apples with high probability even though the predicted probability of “Granny Smith” is near zero. Even more notable, the model never produces pictures of iPods, despite the very high relative predicted probability of this caption.

Another possibility is to probe the structure of the CLIP latent space itself. In Figure 6, we take the CLIP image embeddings of a handful of source images and reconstruct them with progressively more PCA dimensions, and then visualize the reconstructed image embeddings using our decoder with DDIM on a fixed seed. This allows us to see what semantic information the different dimensions encode. We observe that the early PCA dimensions preserve coarse-grained semantic information such as what types of objects are in the scene, whereas the later PCA dimensions encode finer-grained detail such as the shapes and exact form of the objects. For example, in the first scene, the earlier dimensions seem to encode that there is food and perhaps a container present, whereas the later dimensions encode tomatoes and a bottle specifically. Figure 6 also
Granny Smith: 100%
iPod: 0%
Pizza: 0%

Granny Smith: 0.02%
iPod: 99.98%
Pizza: 0%

Granny Smith: 94.33%
iPod: 0%
Pizza: 5.66%

Figure 5: Variations of images featuring typographic attacks [20] paired with the CLIP model’s predicted probabilities across three labels. Surprisingly, the decoder still recovers Granny Smith apples even when the predicted probability for this label is near 0%. We also find that our CLIP model is slightly less susceptible to the “pizza” attack than the models investigated in [20].

serves as a visualization of what the AR prior is modeling, since the AR prior is trained to explicitly predict these principal components in this order.

4 Text-to-Image Generation

4.1 Importance of the Prior

Although we train a prior to generate CLIP image embeddings from captions, the prior is not strictly necessary for caption-to-image generation. For instance, our decoder can condition on both CLIP image embeddings and captions, but the CLIP image embedding is dropped 5% of the time during training in order to enable classifier-free guidance. Therefore, at sampling time, we can condition on only the caption, although this underperforms a model trained fully in this way (this model is GLIDE, and we do a thorough comparison with GLIDE in Sections 4.2 and 4.3). Another possibility is to feed the decoder the CLIP text embedding as if it were an image embedding, as previously observed [61] [54]. The first two rows of Figure 7 depicts samples obtained in these two ways; the third row depicts samples obtained with a prior. Conditioning the decoder on just the caption is clearly worst, but conditioning on text embeddings zero-shot does produce reasonable
results. Building on this observation, another approach would be to train the decoder to condition on CLIP text embeddings [9] instead of CLIP image embeddings (although we would lose the capabilities mentioned in Section 3).

To quantify the effectiveness of these alternate approaches, we train two models: a small decoder conditioned on CLIP text embeddings, and a small unCLIP stack (diffusion prior and decoder). We then compare samples from the text-embedding decoder, samples from the unCLIP stack, and samples obtained from feeding text embeddings to the unCLIP decoder zero-shot, sweeping across guidance scales for all models. We find that these approaches respectively score FIDs of 9.16, 7.99, and 16.55 on a test set, suggesting the unCLIP approach is best. We also run human evaluations comparing the first two settings, sweeping over sampling hyperparameters for each using our human evaluation proxy model (Appendix A). We find that humans prefer the full unCLIP stack 57.0% ± 3.1% of the time for photorealism and 53.1% ± 3.1% of the time for caption similarity.

Given the importance of the prior, it is worth evaluating different approaches to train the prior. We compare both the AR and diffusion priors throughout our experiments. In all cases (Sections 4.2, 4.4, and 4.5), we find that the diffusion prior outperforms the AR prior for comparable model size and reduced training compute.

### 4.2 Human Evaluations

We observe in Figure [1] that unCLIP is capable of synthesizing complex, realistic images. While we can compare sample quality to past models using FID, its not always aligned with human judgment. To better
gauge the generation capabilities of our system, we conduct systematic human evaluations comparing unCLIP to GLIDE for photorealism, caption similarity, and sample diversity.

We follow the protocol of Ramesh et al. [40] and Nichol et al. [35] for the first two evaluations: for photorealism, users are presented with pairs of images and must choose which looks more photorealistic; for caption similarity, users are additionally prompted with a caption, and must choose which image better matches the caption. In both evaluations, there is a third “Not sure” option. For diversity, we propose a new evaluation protocol in which humans are presented with two $4 \times 4$ grids of samples and must choose which is more diverse (with a third option, “Not sure”). For this evaluation, we produce sample grids using 1,000 captions from the MS-COCO validation set, and always compare sample grids for the same caption. Before running human comparisons, we sweep over sampling hyperparameters for each model using a CLIP linear probe trained to be a proxy for human photorealism evaluations (Appendix A). These hyperparameters are fixed across all three types of evaluation.

We present our results in Table 1. In general, the diffusion prior performs better than the AR prior in pairwise comparisons against GLIDE. We find that humans still slightly prefer GLIDE to unCLIP in terms of photorealism, but the gap is very small. Even with similar photorealism, unCLIP is strongly preferred over GLIDE in terms of diversity, highlighting one of its benefits.
Figure 8: Samples when increasing guidance scale for both unCLIP and GLIDE, using the prompt, “A green vase filled with red roses sitting on top of table.” For unCLIP, we fix the latent vectors sampled from the prior, and only vary the guidance scale of the decoder. For both models, we fix the diffusion noise seed for each column. Samples from unCLIP improve in quality (more realistic lighting and shadows) but do not change in content as we increase guidance scale, preserving semantic diversity even at high decoder guidance scales.

4.3 Improved Diversity-Fidelity Trade-off with Guidance

Compared to GLIDE, we qualitatively observe that unCLIP is able to generate more diverse images while leveraging the guidance technique to improve sample quality. To understand why, consider Figure 8 where we increase guidance scale for both GLIDE and unCLIP. For GLIDE, the semantics (camera angle, color, size) converge as we increase guidance scale, whereas for unCLIP the semantic information of the scene is frozen in the CLIP image embedding and therefore does not collapse when guiding the decoder.

In Section 4.2, we observed that unCLIP achieves similar photorealism as GLIDE while maintaining more diversity, but that its caption matching capabilities were slightly worse. It is natural to ask whether GLIDE’s guidance scale can be lowered to obtain the same diversity level as unCLIP while maintaining better caption matching. In Figure 9, we conduct a more careful study of this question by performing human evaluations across several GLIDE guidance scales. We find that GLIDE at guidance scale 2.0 is very close to the photorealism and caption similarity of unCLIP, while still producing less diverse samples.

Finally, in Figure 10, we compute MS-COCO zero-shot FID while sweeping over guidance scale for both unCLIP and GLIDE, finding that guidance hurts the FID of unCLIP much less so than for GLIDE. In this evaluation, we fix the guidance scale of the unCLIP prior and only vary the guidance scale of the decoder. This is another indication that guidance hurts the diversity of GLIDE much more than unCLIP, since FID heavily penalizes non-diverse generations.
4.4 Comparison on MS-COCO

In the text-conditional image generation literature, it has become standard practice to evaluate FID on the MS-COCO [28] validation set. We present results on this benchmark in Table 2. Like GLIDE and DALL-E, unCLIP is not directly trained on the MS-COCO training set, but can still generalize to the validation set zero-shot. We find that, compared to these other zero-shot models, unCLIP achieves a new state-of-the-art FID of 10.39 when sampling with the diffusion prior. In Figure 11, we visually compare unCLIP to various recent text-conditional image generation models on several captions from MS-COCO. We find that, like the other methods, unCLIP produces realistic scenes that capture the text prompts.
<table>
<thead>
<tr>
<th>Model</th>
<th>FID</th>
<th>Zero-shot FID</th>
<th>Zero-shot FID (filt)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AttnGAN (Xu et al., 2017)</td>
<td>35.49</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DM-GAN (Zhu et al., 2019)</td>
<td>32.64</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DF-GAN (Tao et al., 2020)</td>
<td>21.42</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DM-GAN + CL (Ye et al., 2021)</td>
<td>20.79</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XMC-GAN (Zhang et al., 2021)</td>
<td>9.33</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LAFITE (Zhou et al., 2021)</td>
<td></td>
<td>8.12</td>
<td></td>
</tr>
<tr>
<td>Make-A-Scene (Gafni et al., 2022)</td>
<td>7.55</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DALL-E (Ramesh et al., 2021)</td>
<td>~28</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LAFITE (Zhou et al., 2021)</td>
<td>26.94</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GLIDE (Nichol et al., 2021)</td>
<td>12.24</td>
<td>12.89</td>
<td></td>
</tr>
<tr>
<td>Make-A-Scene (Gafni et al., 2022)</td>
<td></td>
<td>11.84</td>
<td></td>
</tr>
<tr>
<td>unCLIP (AR prior)</td>
<td>10.63</td>
<td>11.08</td>
<td></td>
</tr>
<tr>
<td>unCLIP (Diffusion prior)</td>
<td><strong>10.39</strong></td>
<td><strong>10.87</strong></td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Comparison of FID on MS-COCO 256 × 256. For our model, we achieve the best results using the diffusion prior with guidance scale 1.25 for both models.

### 4.5 Aesthetic Quality Comparison

We additionally perform automated aesthetic quality evaluations comparing unCLIP to GLIDE. Our goal with this evaluation is to assess how well each model produces artistic illustrations and photographs. To this end, we generated 512 "artistic" captions using GPT-3 [4] by prompting it with captions for existing artwork (both real and AI generated). Next, we trained a CLIP linear probe to predict human aesthetic judgments using the AVA dataset [33] (Appendix A). For each model and set of sampling hyperparameters, we produce four images for each prompt, and report the mean predicted aesthetic judgment over the full batch of 2048 images.

In Figure 12, we present results on our aesthetic quality evaluation. We find that guidance improves aesthetic quality for both GLIDE and unCLIP. For unCLIP, we only guide the decoder (we found that guiding the prior hurt results). We also plot the aesthetic quality against Recall,

### 5 Related Work

Synthetic image generation is a well studied problem, and most popular techniques for unconditional image generation have also been applied to the text-conditional setting. Many previous works have trained GANs [21] on publicly available image captioning datasets to produce text-conditional image samples [56, 63, 49, 58, 57]. Other works have adapted the VQ-VAE approach [52] to text-conditional image generation by training autoregressive transformers on sequences of text tokens followed by image tokens [40, 12, 1]. Finally, some works have applied diffusion models to the problem, training either continuous [35] or discrete [22] diffusion models with auxiliary text encoders to handle textual input.

Previous works have leveraged hierarchical generative processes to create high-quality synthetic images. Razavi et al. [41] trains a multi-layer discrete autoencoder, allowing them to first sample coarse-grained latent codes and then use this as conditioning information when sampling higher-resolution latent codes. Child, Vahdat and Kautz [5, 50] generate images using VAEs with a hierarchy of latent codes that increase progressively with resolution. Concurrently with our work, Gafni et al. [17] conditions a generative image

Recall is computed with respect to the training dataset.
model on segmentation masks, allowing for a generative process that first samples a semantic map of an image and then conditions the generated image on this information.

The computational benefits of using diffusion to model a latent space has been noted by previous works. Preechakul et al. [38] propose an autoencoder framework where diffusion models are used to render latent variables as images, and a second diffusion model is used to generate these latents (similar to our diffusion model on segmentation masks, allowing for a generative process that first samples a semantic map of an image and then conditions the generated image on this information.

The computational benefits of using diffusion to model a latent space has been noted by previous works. Preechakul et al. [38] propose an autoencoder framework where diffusion models are used to render latent variables as images, and a second diffusion model is used to generate these latents (similar to our diffusion model on segmentation masks, allowing for a generative process that first samples a semantic map of an image and then conditions the generated image on this information.

The computational benefits of using diffusion to model a latent space has been noted by previous works. Preechakul et al. [38] propose an autoencoder framework where diffusion models are used to render latent variables as images, and a second diffusion model is used to generate these latents (similar to our diffusion model on segmentation masks, allowing for a generative process that first samples a semantic map of an image and then conditions the generated image on this information.
Figure 12: Aesthetic quality evaluations comparing GLIDE and unCLIP using 512 auto-generated artistic prompts. We find that both models benefit from guidance, but unCLIP does not sacrifice recall for aesthetic quality.

prior). Vahdat et al. \[51\] use a score-based model for the latent space of a VAE, while Rombach et al. \[42\] use diffusion models on the latents obtained from a VQGAN \[14\] like autoencoder.

Since its release, CLIP \[39\] has been used extensively to steer generative image models towards text prompts. Galatolo et al., Patashnik et al., Murdock, Gal et al. \[19, 36, 32, 18\] guide GANs using gradients from a CLIP model. For diffusion models, Dhariwal and Nichol \[11\] introduced classifier guidance as a way to use gradients from a classifier trained on noised images to steer the model towards higher quality generations. Nichol et al. \[35\] train a CLIP model on noised images and guide a text-conditional diffusion model, while Crowson \[7, 8\] use an unnoised CLIP model to guide unconditional or class-conditional diffusion models. Ho and Salimans \[24\] introduced classifier-free guidance and showed that one can perform guidance implicitly from the predictions of the model with and without the conditioning information, thus removing the need for a classifier. Nichol et al. \[35\] showed classifier-free guidance works more favorably than CLIP guidance for text conditional image generation.

Several previous works have trained generative image models that are directly conditioned on CLIP embeddings. Zhou et al. \[61\] condition GAN models on randomly perturbed CLIP image embeddings, finding that these models can generalize to CLIP text embeddings to produce text-conditional images. Crowson \[9\] trained diffusion models conditioned on CLIP text embeddings, allowing for direct text-conditional image generation. Wang et al. \[54\] train an autoregressive generative model conditioned on CLIP image embeddings, finding that it generalizes to CLIP text embeddings well enough to allow for text-conditional image synthesis.

Bordes et al. \[3\] train diffusion models conditioned on image representations from contrastive models. While the diffusion models themselves cannot generate images unconditionally, the authors experimented with a simple approach for two-stage image generation by employing Kernel Density Estimation to sample image representations. By feeding these generated representations to the diffusion model, they can generate images end-to-end in a way similar to our proposed technique. However, our work differs from this in two ways: first, we use multimodal contrastive representations rather than image-only representations; second, we employ much more powerful generative models for the first stage of the generation hierarchy, and these generative models are conditioned on text.

6 Limitations and Risks

Although conditioning image generation on CLIP embeddings improves diversity, this choice does come with certain limitations. In particular, unCLIP is worse at binding attributes to objects than a corresponding GLIDE
Figure 13: Samples from unCLIP and GLIDE for the prompt “a red cube on top of a blue cube”.

Figure 14: Reconstructions from the decoder for difficult binding problems. We find that the reconstructions mix up objects and attributes. In the first two examples, the model mixes up the color of two objects. In the rightmost example, the model does not reliably reconstruct the relative size of two objects.

model. In Figure 13 we find that unCLIP struggles more than GLIDE with a prompt where it must bind two separate objects (cubes) to two separate attributes (colors). We hypothesize that this occurs because the CLIP embedding itself does not explicitly bind attributes to objects, and find that reconstructions from the decoder
Figure 15: Samples from unCLIP for the prompt, “A sign that says deep learning.”

(a) A high quality photo of a dog playing in a green field next to a lake.

(b) A high quality photo of Times Square.

Figure 16: unCLIP samples showing low levels of detail for some complex scenes.
often mix up attributes and objects, as shown in Figure [14]. A similar and likely related issue is that unCLIP struggles at producing coherent text, as illustrated in Figure [15] it is possible that the CLIP embedding does not precisely encode spelling information of rendered text. This issue is likely made worse because the BPE encoding we use obscures the spelling of the words in a caption from the model, so the model needs to have independently seen each token written out in the training images in order to learn to render it.

We also note that our stack still has a hard time producing details in complex scenes (Figure [16]). We hypothesize that this is a limitation of our decoder hierarchy producing an image at a base resolution of $64 \times 64$ and then upsampling it. Training our unCLIP decoder at a higher base resolution should be able to alleviate this, at the cost of needing more training and inference compute.

As also discussed in the GLIDE paper, image generation models carry risks related to deceptive and otherwise harmful content. unCLIP’s performance improvements also raise the risk profile over GLIDE. As the technology matures, it leaves fewer traces and indicators that outputs are AI-generated, making it easier to mistake generated images for authentic ones and vice versa. More research is also needed on how the change in architecture changes how the model learns biases in training data.

The risks of these models should be assessed in relation to the particular deployment context – which include training data, guardrails in place, the deployment space, and who will have access. A preliminary analysis of these issues in the context of the DALL·E 2 Preview platform (the first deployment of an unCLIP model), can be found in [Mishkin et al.] [30].

7 Acknowledgements

We’d like to thank Jong Wook Kim, Hyeonwoo Noh, Alec Radford, Pranav Shyam, and Ilya Sutskever for helpful discussions and contributions to our work. We’d also like to thank Yunxin Jiao for creating several figures used in the paper. We are grateful to the Acceleration and Supercomputing teams at OpenAI for their work on software and hardware infrastructure this project used.

References


A Linear Probes for Evaluations

For our evaluations, we leverage two new linear probes on top of a CLIP ViT-L/14 model. To automate aesthetic quality evaluations, we follow the procedure used by Crowson [6], training a linear regression model on images and mean ratings from the AVA dataset [33]. To reduce the cost of hyperparameter sweeps before conducting human evaluations, we train a logistic regression model to predict win probabilities between pairs of images. To train this model, we used 15,000 pairwise image comparisons gathered from all of our previous human evaluations. For each comparison \( i \), we computed CLIP image embeddings \( x_i \) and \( y_i \) for the two images in the pair. We then trained a linear model \( f(x) \) such that \( 1/(1 + \exp(f(x_i) - f(y_i))) \) approximates the probability that a human prefers the image for \( y_i \). This can be reduced to a logistic regression problem with inputs equal to \( y_i - x_i \).

B Error Bars for Human Evaluation

When computing error bars for human evaluations, we use the normal approximation interval with \( p = 0.95 \). We expect the normal approximation to be accurate for such a large sample size of \( n = 1000 \).

C Training details

We report model and training hyperparameters in Table 3. All models were trained using Adam [27] with corrected weight decay [29] and momentum \( \beta_1 = 0.9 \).

Our CLIP model uses a ViT-H/16 [13] image encoder that consumes 256 \( \times \) 256 resolution images, and has width 1280 with 32 Transformer [53] blocks. The text encoder also follows the architecture described in Radford et al. [39]: it is a Transformer [53] with a causal attention mask, with width 1024 and 24 Transformer blocks. Both models are trained with learning rate \( 3 \times 10^{-4} \) and SAM [15] with \( \rho = 0.1 \), where the perturbations are applied independently by the replicas, each of which uses batch size 64. The remaining hyperparameters are the same as those reported in Radford et al. [39].

When training the encoder, we sample from the CLIP [39] and DALL-E [40] datasets (approximately 650M images in total) with equal probability. When training the decoder, upsamplers, and prior, we use only the DALL-E dataset [40] (approximately 250M images). Incorporating the noisier CLIP dataset while training the generative stack negatively impacted sample quality in our initial evaluations.

Our decoder architecture is the 3.5 billion parameter GLIDE model, with the same architecture and diffusion hyperparameters as in Nichol et al. [35]. We train with learned sigma and sample with 250 strided sampling steps as in Nichol and Dhariwal [34].

We use the ADMNet architecture [11] for the upsamplers. In the first upsampling stage, we use a cosine noising schedule, 320 channels and a depth of 3 resblocks per resolution inside the ADMNet. We also apply gaussian blur (kernel size 3, sigma 0.6) as described in Saharia et al. [43]. In the second upsampling stage, we use a linear noising schedule, 192 channels, a depth of 2 resblocks per resolution, and train with the BSR degradation from Rombach et al. [42]. Neither upsampler uses attention. To reduce inference time, we use DDIM [47] and manually tune the number of steps, with 27 steps for 256 \( \times \) 256 model, and 15 steps for the 1024 \( \times \) 1024 model.

For the AR prior, we use a Transformer text encoder with width 2048 and 24 blocks and a decoder with a causal attention mask, width 1664, and 24 blocks. For the diffusion prior, we use a Transformer with width 2048 and 24 blocks, and sample with Analytic DPM [2] with 64 strided sampling steps. To reuse hyperparameters tuned for diffusion noise schedules on images from Dhariwal and Nichol [11], we scale the CLIP embedding inputs by 17.2 to match the empirical variance of RGB pixel values of ImageNet images scaled to \([-1, 1]\).
<table>
<thead>
<tr>
<th></th>
<th>AR prior</th>
<th>Diffusion prior</th>
<th>64</th>
<th>64 → 256</th>
<th>256 → 1024</th>
</tr>
</thead>
<tbody>
<tr>
<td>Diffusion steps</td>
<td>-</td>
<td>1000</td>
<td>1000</td>
<td>1000</td>
<td>1000</td>
</tr>
<tr>
<td>Noise schedule</td>
<td>-</td>
<td>cosine</td>
<td>cosine</td>
<td>cosine</td>
<td>linear</td>
</tr>
<tr>
<td>Sampling steps</td>
<td>-</td>
<td>64</td>
<td>250</td>
<td>27</td>
<td>15</td>
</tr>
<tr>
<td>Crop fraction</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0.25</td>
<td>0.25</td>
</tr>
<tr>
<td>Model size</td>
<td>1B</td>
<td>1B</td>
<td>3.5B</td>
<td>700M</td>
<td>300M</td>
</tr>
<tr>
<td>Channels</td>
<td>-</td>
<td>-</td>
<td>512</td>
<td>320</td>
<td>192</td>
</tr>
<tr>
<td>Depth</td>
<td>-</td>
<td>-</td>
<td>3</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>Channels multiple</td>
<td>-</td>
<td>-</td>
<td>1,2,3,4</td>
<td>1,2,3,4</td>
<td>1,1,2,2,4,4</td>
</tr>
<tr>
<td>Heads channels</td>
<td>-</td>
<td>-</td>
<td>64</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Attention resolution</td>
<td>-</td>
<td>-</td>
<td>32,16,8</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Text encoder context</td>
<td>256</td>
<td>256</td>
<td>256</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Text encoder width</td>
<td>2048</td>
<td>2048</td>
<td>2048</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Text encoder depth</td>
<td>24</td>
<td>24</td>
<td>24</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Text encoder heads</td>
<td>32</td>
<td>32</td>
<td>32</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Latent decoder context</td>
<td>384</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Latent decoder width</td>
<td>1664</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Latent decoder depth</td>
<td>24</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Latent decoder heads</td>
<td>26</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Dropout</td>
<td>-</td>
<td>-</td>
<td>0.1</td>
<td>0.1</td>
<td>-</td>
</tr>
<tr>
<td>Weight decay</td>
<td>4.0e-2</td>
<td>6.0e-2</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Batch size</td>
<td>4096</td>
<td>4096</td>
<td>2048</td>
<td>1024</td>
<td>512</td>
</tr>
<tr>
<td>Iterations</td>
<td>1M</td>
<td>600K</td>
<td>800K</td>
<td>1M</td>
<td>1M</td>
</tr>
<tr>
<td>Learning rate</td>
<td>1.6e-4</td>
<td>1.1e-4</td>
<td>1.2e-4</td>
<td>1.2e-4</td>
<td>1.0e-4</td>
</tr>
<tr>
<td>Adam $\beta_2$</td>
<td>0.91</td>
<td>0.96</td>
<td>0.999</td>
<td>0.999</td>
<td>0.999</td>
</tr>
<tr>
<td>Adam $\epsilon$</td>
<td>1.0e-10</td>
<td>1.0e-6</td>
<td>1.0e-8</td>
<td>1.0e-8</td>
<td>1.0e-8</td>
</tr>
<tr>
<td>EMA decay</td>
<td>0.999</td>
<td>0.999</td>
<td>0.9999</td>
<td>0.9999</td>
<td>0.9999</td>
</tr>
</tbody>
</table>

Table 3: Hyperparameters for the models
In Figures 17, 18 and 19 we show random samples from our production model for some of the prompts from Figure 1.

Figure 17: Random samples from unCLIP for prompt “Vibrant portrait painting of Salvador Dali with a robotic half face”
Figure 18: Random samples from unCLIP for prompt “A close up of a handpalm with leaves growing from it.”
Figure 19: Random samples from unCLIP for prompt “A teddybear on a skateboard in Times Square.”