Scaling up GANs for Text-to-Image Synthesis

Minguk Kang\(^1,3\) Jun-Yan Zhu\(^2\) Richard Zhang\(^3\)
Jaesik Park\(^1\) Eli Shechtman\(^3\) Sylvain Paris\(^3\) Taesung Park\(^3\)
\(^1\)POSTECH \(^2\)Carnegie Mellon University \(^3\)Adobe Research

Abstract

The recent success of text-to-image synthesis has taken the world by storm and captured the general public’s imagination. From a technical standpoint, it also marked a drastic change in the favored architecture to design generative image models. GANs used to be the de facto choice, with techniques like StyleGAN. With DALL-E 2, autoregressive and diffusion models became the new standard for large-scale generative models overnight. This rapid shift raises a fundamental question: can we scale up GANs to benefit from large datasets like LAION? We find that na"ively increasing the capacity of the StyleGAN architecture quickly becomes unstable. We introduce GigaGAN, a new GAN architecture that far exceeds this limit, demonstrating GANs as a viable option for text-to-image synthesis. GigaGAN offers three major advantages. First, it is orders of magnitude faster at inference time, taking only 0.13 seconds to synthesize a 512px image. Second, it can synthesize high-resolution images, for example, 16-megapixel images in 3.66 seconds. Finally, GigaGAN supports various latent space editing applications such as latent interpolation, style mixing, and vector arithmetic operations.

1. Introduction

Recently released models, such as DALL-E 2 [74], Imagen [80], Parti [101], and Stable Diffusion [79], have ushered in a new era of image generation, achieving unprecedented levels of image quality and model flexibility. The now-dominant paradigms, diffusion models and autoregressive models, both rely on iterative inference. This is a double-edged sword, as iterative methods enable stable training with simple objectives but incur a high computational cost during inference.

Contrast this with Generative Adversarial Networks (GANs) [6,21,41,72], which generate images through a single forward pass and thus inherently efficient. While such models dominated the previous “era” of generative modeling, scaling them requires careful tuning of the network architectures and training considerations due to instabilities in the training procedure. As such, GANs have excelled at modeling single or multiple object classes, but scaling to complex datasets, much less an open world, has remained challenging. As a result, ultra-large models, data, and compute resources are now dedicated to diffusion and autoregressive models. In this work, we ask – can GANs continue to be scaled up and potentially benefit from such resources, or have they plateaued? What prevents them from further scaling, and can we overcome these barriers?

We first experiment with StyleGAN2 [42] and observe that simply scaling the backbone causes unstable training. We identify several key issues and propose techniques to stabilize the training while increasing the model capacity. First, we effectively scale the generator’s capacity by retaining a bank of filters and taking a sample-specific linear combination. We also adapt several techniques commonly used in the diffusion context and confirm that they bring similar benefits to GANs. For instance, interleaving both self-attention (image-only) and cross-attention (image-text) with the convolutional layers improves performance.

Furthermore, we reintroduce multi-scale training, finding a new scheme that improves image-text alignment and low-frequency details of generated outputs. Multi-scale training allows the GAN-based generator to use parameters in low-resolution blocks more effectively, leading to better image-text alignment and image quality. After careful tuning, we achieve stable and scalable training of a one-billion-parameter GAN (GigaGAN) on large-scale datasets, such as LAION2B-en [88]. Our results are shown in Figure 1.

In addition, our method uses a multi-stage approach [14, 104]. We first generate at 64 × 64 and then upsample to 512 × 512. These two networks are modular and robust enough to be used in a plug-and-play fashion. We show that our text-conditioned GAN-based upsampling network can be used as an efficient, higher-quality upsampler for a base diffusion model such as DALL-E 2, despite never having seen diffusion images at training time (Figures 2).

Together, these advances enable our GigaGAN to go far beyond previous GANs: \(36 \times\) larger than Style-
Figure 1. Our model, GigaGAN, shows GAN frameworks can also be scaled up for general text-to-image synthesis tasks, generating a 512px output at an interactive speed of 0.13s, and 4096px at 3.7s. Selected examples at 2K or 4K resolutions are shown. Please zoom in for more details. See Appendix C and our website for more uncurated comparisons.
Figure 2. **Our GAN-based upsampler** can serve in the upsampling pipeline of many text-to-image models that often generate initial outputs at low resolutions like 64px or 128px. We simulate such usage by applying our text-conditioned 8× superresolution model on a low-res 128px artwork to obtain the 1K output, using “Portrait of a colored iguana dressed in a hoodie”. Then our model can be re-applied to go beyond 4K. We compare our model with the text-conditioned upscaler of Stable Diffusion [78] and unconditional Real-ESRGAN [33]. Zooming in is recommended for comparison between 1K and 4K.
Figure 3. Our GAN-based upsampler, similar to Figure 2, can also be used as an off-the-shelf superresolution model for real images with a large scaling factor by providing an appropriate description of the image. We apply our text-conditioned 8× superresolution model on a low-res 128px photo to obtain the 1K output, using “A dog sitting in front of a mini tipi tent”. Then our model can be re-applied to go beyond 4K. We compare our model with the text-conditioned upscaler of Stable Diffusion [78] and unconditional Real-ESRGAN [33]. Zooming in is recommended for comparison between 1K and 4K.
GAN2 [42] and 6× larger than StyleGAN-XL [86] and XMC-GAN [103]. While our 1B parameter count is still lower than the largest recent synthesis models, such as imagen (3.0B), DALL-E 2 (5.5B), and Parti (20B), we have not yet observed a quality saturation regarding the model size. GigaGAN achieves a zero-shot FID of 9.09 on COCO2014 dataset, lower than the FID of DALL-E 2, Parti-750M, and Stable Diffusion.

Furthermore, GigaGAN has three major practical advantages compared to diffusion and autoregressive models. First, it is orders of magnitude faster, generating a 512px image in 0.13 seconds (Figure 1). Second, it can synthesize ultra high-res images at 4k resolution in 3.66 seconds. Third, it is endowed with a controllable, latent vector space that lends itself to well-studied controllable image synthesis applications, such as style mixing (Figure 6), prompt interpolation (Figure 7), and prompt mixing (Figure 8).

In summary, our model is the first GAN-based method that successfully trains a billion-scale model on billions of real-world complex Internet images. This suggests that GANs are still a viable option for text-to-image synthesis and should be considered for future aggressive scaling. Please visit our website for additional results.

2. Related Works

Text-to-image synthesis. Generating a realistic image given a text description, explored by early works [58, 113], is a challenging task. A common approach is text-conditional GANs [76, 77, 93, 99, 104, 111] on specific domains [96] and datasets with a closed-world assumption [54]. With the development of diffusion models [15, 26], autoregressive (AR) transformers [12], and large-scale language encoders [71, 73], text-to-image synthesis has shown remarkable improvement on an open-world of arbitrary text descriptions. GLIDE [63], DALL-E 2 [74], and Imagen [80] are representative diffusion models that show photorealistic outputs with the aid of a pretrained language encoder [71, 73]. AR models, such as DALL-E [75], Make-A-Scene [20], CogView [16, 17], and Parti [101] also achieve amazing results. While these models exhibit unprecedented image synthesis ability, they require time-consuming iterative processes to achieve high-quality image sampling.

To accelerate the sampling, several methods propose to reduce the sampling steps [57, 59, 83, 89] or reuse pre-computed features [51]. Latent Diffusion Model (LDM) [79] performs the reverse processes in low-dimensional latent space instead of pixel space. However, consecutive reverse processes are still computationally expensive, limiting the usage of large-scale text-to-image models for interactive applications.

GAN-based image synthesis. GANs [21] have been one of the primary families of generative models for natural image synthesis. As the sampling quality and diversity of GANs improve [39–42, 44, 72, 84], GANs have been deployed to various computer vision and graphics applications, such as text-to-image synthesis [76], image-to-image translation [29, 34, 49, 65, 66, 110], and image editing [1, 7, 69, 109]. Notably, StyleGAN-family models [40, 42] have shown impressive ability in image synthesis tasks for single-category domains [1, 31, 69, 98, 112]. Other works have explored class-conditional GANs [6, 36, 86, 102, 107] on datasets with a fixed set of object categories.

In this paper, we change the data regimes from single- or multi-categories datasets to extremely data-rich situations. We make the first expedition toward training a large-scale GAN for text-to-image generation on a vast amount of web-crawled text and image pairs, such as LAION2B-en [88] and COYO-700M [8]. Existing GAN-based text-to-image synthesis models [52, 76, 93, 99, 103, 104, 111] are trained on relatively small datasets, such as CUB-200 (12k training pairs), MSCOCO (82k) and LN-OpenImages (507k). Also, those models are evaluated on associated validation datasets, which have not been validated to perform large-scale text-image synthesis like diffusion or AR models.

Concurrent with our method, StyleGAN-T [85] and GALIP [92] share similar goals and make complementary insights to ours.

Super-resolution for large-scale text-to-image models. Large-scale models require prohibitive computational costs for both training and inference. To reduce the memory and running time, cutting-edge text-to-image models [63, 74, 80, 101] have adopted cascaded generation processes where images are first generated at $64 \times 64$ resolution and upsampled to $256 \times 256$ and $1024 \times 1024$ sequentially. However, the super-resolution networks are primarily based on diffusion models, which require many iterations. In contrast, our low-res image generators and upsamplers are based on GANs, reducing the computational costs for both stages. Unlike traditional super-resolution techniques [2, 18, 47, 95] that aim to faithfully reproduce low-resolution inputs or handle image degradation like compression artifacts, our upsamplers for large-scale models serve a different purpose. They need to perform larger upsampling factors while potentially leveraging the input text prompt.

3. Method

We train a generator $G(z, c)$ to predict an image $x \in \mathbb{R}^{H\times W\times 3}$ given a latent code $z \sim N(0, 1) \in \mathbb{R}^{128}$ and text-conditioning signal $c$. We use a discriminator $D(x, c)$ to judge the realism of the generated image, as compared to a sample from the training database $D$, which contains image-text pairs.
Although GANs [6, 39, 41] can successfully generate realistic images on single- and multi-category datasets [13, 41, 100], open-ended text-conditioned synthesis on Internet images remains challenging. We hypothesize that the current limitation stems from its reliance on convolutional layers. That is, the same convolution filters are challenged to model the general image synthesis function for all text conditioning and by capturing long-range dependence via the attention mechanism.

Below, we discuss our key contributions to making ConvNets more expressive (Section 3.1), followed by our designs for the generator (Section 3.2) and discriminator (Section 3.3). Lastly, we introduce a new, fast GAN-based upsampler model that can improve the inference quality and speed of our method and diffusion models such asImagen [80] and DALL-E 2 [74].

### 3.1. Modeling complex contextual interaction

#### Baseline StyleGAN generator. **We base our architecture off the conditional version of StyleGAN2 [42], comprised of two networks** $G = \hat{G} \circ M$. The mapping network $w = M(z, c)$ maps the inputs into a “style” vector $w$, which modulates a series of upsampling convolutional layers in the synthesis network $\hat{G}(w)$ to map a learned constant tensor to an output image $x$. Convolution is the main engine to generate all output pixels, with the $w$ vector as the only source of information to model conditioning.

#### Sample-adaptive kernel selection. **To handle the highly diverse distribution of internet images, we aim to increase the capacity of convolution kernels. However, increasing the width of the convolution layers becomes too demanding, as the same operation is repeated across all locations.**

We propose an efficient way to enhance the expressivity of convolutional kernels by creating them on-the-fly based on the text conditioning, as illustrated in Figure 4 (right). In this scheme, we instantiate a bank of $N$ filters $\{K_i \in \mathbb{R}^{C_{in} \times C_{out} \times K \times K}\}_{i=1}^{N}$, instead of one, that takes a feature $f \in \mathbb{R}^{C_{in}}$ at each layer. The style vector $w \in \mathbb{R}^{d}$ then goes through an affine layer $[W_{\text{filter}}, b_{\text{filter}}] \in \mathbb{R}^{(d+1) \times N}$ to predict a set of weights to average across the filters, to produce an aggregated filter $K \in \mathbb{R}^{C_{in} \times C_{out} \times K \times K}$.

$$K = \sum_{i=1}^{N} K_i \cdot \text{softmax}(W_{\text{filter}}^T w + b_{\text{filter}})_i$$  \hspace{1cm} (1)

The filter is then used in the regular convolution pipeline of StyleGAN2, with the second affine layer $[W_{\text{mod}}, b_{\text{mod}}] \in \mathbb{R}^{(d+1) \times C_{in}}$ for weight (de-)modulation [42].

$$g_{\text{adaconv}}(f, w) = (W_{\text{mod}}^T w + b_{\text{mod}}) \odot K \ast f$$  \hspace{1cm} (2)

where $\odot$ and $\ast$ represent (de-)modulation and convolution.

At a high level, the softmax-based weighting can be viewed as a differentiable filter selection process based on input conditioning. Furthermore, since the filter selection process is performed only once at each layer, the selection process is much faster than the actual convolution, decoupling compute complexity from the resolution. Our method...
shares a spirit with dynamic convolutions \[23, 35, 91, 97\] in that the convolution filters dynamically change per sample, but differs in that we explicitly instantiate a larger filter bank and select weights based on a separate pathway conditional on the w-space of StyleGAN.

**Interleaving attention with convolution.** Since the convolutional filter operates within its receptive field, it cannot contextualize itself in relationship to distant parts of the images. One way to incorporate such long-range relationships is using attention layers. While recent diffusion-based models \[15, 27, 79\] have commonly adopted attention mechanisms, StyleGAN architectures are predominantly convolutional with the notable exceptions such as BigGAN \[6\], GANformer \[30\], and ViTGAN \[50\].

We aim to improve the performance of StyleGAN by integrating attention layers with the convolutional backbone. However, simply adding attention layers to StyleGAN often results in training collapse, possibly because the dot-product self-attention is not Lipschitz, as pointed out by Kim et al. \[43\]. As the Lipschitz continuity of discriminators has played a critical role in stable training \[3, 22, 60\], we use the L2-distance instead of the dot product as the attention logits to promote Lipschitz continuity \[43\], similar to ViTGAN \[50\].

To further improve performance, we find it crucial to match the architectural details of StyleGAN, such as equalized learning rate \[39\] and weight initialization from a unit normal distribution. We scale down the L2 distance logits roughly to match the unit normal distribution at initialization and reduce the residual gain from the attention layers. We further improve stability by tying the key and query matrix \[50\], and applying weight decay.

In the synthesis network \(G\), the attention layers are interleaved with each convolutional block, leveraging the style vector \(w\) as an additional token. At each attention block, we add a separate cross-attention mechanism \(g_{\text{cross-attention}}\) to attend to individual word embeddings \[4\]. We use each input feature tensor as the query, and the text embeddings as the key and value of the attention mechanism.

**3.2. Generator design**

**Text and latent-code conditioning.** First, we extract the text embedding from the prompt. Previous works \[75, 80\] have shown that leveraging a strong language model is essential for producing strong results. To do so, we tokenize the input prompt (after padding it to \(C = 77\) words, following best practices \[75, 80\]) to produce conditioning vector \(c \in \mathbb{R}^{C \times 768}\), and take the features from the penultimate layer \[80\] of a frozen CLIP feature extractor \[71\]. To allow for additional flexibility, we apply additional attention layers \(T\) on top to process the word embeddings before passing them to the MLP-based mapping network. This results in text embedding \(t = T(c_{\text{txt}}(c)) \in \mathbb{R}^{C \times 768}\). Each component \(t_i\) of \(t\) captures the embedding of the \(i\)th word in the sentence. We refer to them as \(t_{\text{local}} = t_{\{1:C\} \setminus \text{EOT}} \in \mathbb{R}^{(C-1) \times 768}\). The EOT (“end of text”) component of \(t\) aggregates global information, and is called \(t_{\text{global}} \in \mathbb{R}^{768}\).

We process this global text descriptor, along with the latent code \(z \sim \mathcal{N}(0, 1)\), via an MLP mapping network to extract the style \(w = M(z, t_{\text{global}})\).

\[
(t_{\text{local}}, t_{\text{global}}) = T(c_{\text{txt}}(c)),
\]
\[
w = M(z, t_{\text{global}}).
\] (3)

Different from the original StyleGAN, we use both the text-based style code \(w\) to modulate the synthesis network \(G\) and the word embeddings \(t_{\text{local}}\) as features for cross-attention.

\[
x = G(w, t_{\text{local}}).
\] (4)

Similar to earlier works \[58, 74, 80\], the text-image alignment visually improves with cross-attention.

**Synthesis network.** Our synthesis network consists of a series of upsampling convolutional layers, with each layer enhanced with the adaptive kernel selection (Equation 1) and followed by our attention layers.

\[
f_{\ell+1} = g_{\text{attn}}^\ell(g_{\text{attn}}^\ell(g_{\text{attn}}^\ell(f_{\ell}, w), w), t_{\text{local}}),
\] (5)

where \(g_{\text{attn}}^\ell, g_{\text{attn}}^\ell,\) and \(g_{\text{attn}}^\ell\) denote the \(l\)-th layer of cross-attention, self-attention, and weight (de-)modulation layers. We find it beneficial to increase the depth of the network by adding more blocks at each layer. In addition, our generator outputs a multi-scale image pyramid with \(L = 5\) levels, instead of a single image at the highest resolution, similar to MSG-3GAN \[38\] and AnyCostGAN \[53\]. We refer to the pyramid as \(\{x_i\}_{i=0}^{L-1} = \{x_0, x_1, \ldots, x_L\}\), with spatial resolutions \(\{S_i\}_{i=0}^{L-1} = \{64, 32, 16, 8, 4\}\), respectively. The base level \(x_0\) is the output image \(x\). Each image of the pyramid is independently used to compute the GAN loss, as discussed in Section 3.3. We follow the findings of StyleGAN-XL \[86\] and turn off the style mixing and path length regularization \[42\]. We include more training details in Appendix A.1.

**3.3. Discriminator design**

As shown in Figure 5, our discriminator consists of separate branches for processing text with the function \(f_T\) and images with function \(\phi\). The prediction of real vs fake is made by comparing the features from the two branches using function \(\psi\). We introduce a new way of making predictions on multiple scales. Finally, we use additional CLIP and Vision-Aided GAN losses \[44\] to improve stability.
Text conditioning. First, to incorporate conditioning into discriminators, we extract text descriptor $t_D$ from text $c$. Similar to the generator, we apply a pretrained text encoder, such as CLIP [71], followed by a few learnable attention layers. In this case, we only use the global descriptor.

Multiscale image processing. We observe that the early, low-resolution layers of the generator become inactive, using small dynamic ranges irrespective of the provided prompts. StyleGAN2 [42] also observes this phenomenon, concluding that the network relies on the high-resolution layers, as the model size increases. As recovering performance in low frequencies, which contains complex structure information, is crucial, we redesign the model architecture to provide training signals across multiple scales.

Recall the generator produces a pyramid $\{x_i\}_{i=0}^{L-3}$, with the full image $x_0$ at the pyramid base. MSG-GAN [38] improves performance by making a prediction on the entire pyramid at once, enforcing consistency across scales. However, in our large-scale setting, this harms stability, as this pyramid at once, enforcing consistency across scales. How-ever, during early training, when artifacts are obvious, the discriminator heavily relies on making a decision independent of conditioning and hesitates to account for the conditioning.

To enforce the discriminator to incorporate conditioning, we match $x$ with a random, independently sampled condition $c$, and present them as a fake pair:

$$V_{\text{match}} = \mathbb{E}_{x,c,\hat{c}} \left[ \log(1 + \exp(D(x, \hat{c}))) + \log(1 + \exp(D(G(c), \hat{c}))) \right],$$

where $(x, c)$ and $(\hat{c})$ are separately sampled from $p_{\text{data}}$. This loss has previously been explored in text-to-image GAN works [76,104], except we find that enforcing the Matching-aware loss on generated images from $G$, as well real images $x$, leads to clear gains in performance (Table 1).
CLIP contrastive loss. We further leverage off-the-shelf pretrained models as a loss function [44, 84, 90]. In particular, we enforce the generator to produce outputs that are identifiable by the pre-trained CLIP image and text encoders [71], \( \mathcal{E}_{\text{img}} \) and \( \mathcal{E}_{\text{txt}} \), in the contrastive cross-entropy loss that was used to train them originally.

\[
\mathcal{L}_{\text{CLIP}} = \mathbb{E}_{\{c_n\}} \left[ - \log \frac{\exp(\mathcal{E}_{\text{img}}(G(c_0))^\top \mathcal{E}_{\text{txt}}(c_0))}{\sum_n \exp(\mathcal{E}_{\text{img}}(G(c_0))^\top \mathcal{E}_{\text{txt}}(c_n))} \right],
\]

where \( \{c_n\} = \{c_0, \ldots\} \) are sampled captions from the training data.

Vision-aided adversarial loss. Lastly, we build an additional discriminator that uses the CLIP model as a backbone, known as Vision-Aided GAN [44]. We freeze the CLIP image encoder, extract features from the intermediate layers, and process them through a simple network with \( 3 \times 3 \) conv layers to make real/fake predictions. We also incorporate conditioning through modulation, as in Equation 7. To stabilize training, we also add a fixed random projection layer, as proposed by Projected GAN [84]. We refer to this as \( \mathcal{L}_{\text{vision}}(G) \) (omitting the learnable discriminator parameters for clarity).

Our final objective is \( \mathcal{V}(G, D) = \mathcal{V}_{\text{MS-IO}}(G, D) + \mathcal{L}_{\text{CLIP}}(G) + \mathcal{L}_{\text{vision}}(G) \), with weighting between the terms specified in Table A2.

3.4. GAN-based upsampler

Furthermore, GigaGAN framework can be easily extended to train a text-conditioned superresolution model, capable of upsampling the outputs of the base GigaGAN generator to obtain high-resolution images at 512px or 2k resolution. By training our pipeline in two separate stages, we can afford a higher capacity 64px base model within the same computational resources.

In the upsampler, the synthesis network is rearranged to an asymmetric U-Net architecture, which processes the 64px input through 3 downsampling residual blocks, followed by 6 upsampling residual blocks with attention layers to produce the 512px image. There exist skip connections at the same resolution, similar to CoModGAN [106]. The model is trained with the same losses as the base model, as well as the LPIPS Perceptual Loss [105] with respect to the ground truth high-resolution image. Vision-aided GAN is not used for the upsampler. During training and inference time, we apply moderate Gaussian noise augmentation to reduce the gap between real and GAN-generated images. Please refer to Appendix A.3 for more details.

Our GigaGAN framework becomes particularly effective for the superresolution task compared to the diffusion-based models, which cannot afford as many sampling steps as the base model at high resolution. The LPIPS regression loss also provides a stable learning signal. We believe that our GAN upsampler can serve as a drop-in replacement for the superresolution stage of other generative models.

4. Experiments

Systematic, controlled evaluation of large-scale text-to-image synthesis tasks is difficult, as most existing models are not publicly available. Training a new model from scratch would be prohibitively costly, even if the training code were available. Still, we compare our model to recent text-to-image models, such as Imagen [80], Latent Diffusion Models (LDM) [79], Stable Diffusion [78], and Parti [101], based on the available information, while acknowledging considerable differences in the training dataset, number of iterations, batch size, and model size. In addition to text-to-image results, we evaluate our model on ImageNet class-conditional generation in Appendix B, for an apples-to-apples comparison with other methods at a more controlled setting.

For quantitative evaluation, we mainly use the Fréchet Inception Distance (FID) [25] for measuring the realism of the output distribution and the CLIP score for evaluating the image-text alignment.

We conduct five different experiments. First, we show the effectiveness of our method by gradually incorporating each technical component one by one (Section 4.2). Second, our text-to-image synthesis results demonstrate that GigaGAN exhibits comparable FID with Stable Diffusion (SD-v1.5) [79] while generating results hundreds of times faster than diffusion or autoregressive models (Section 4.3). Third, we compare GigaGAN with a distillation-based diffusion model [59] and show that GigaGAN can synthesize higher-quality images faster than the distillation-based diffusion model. Fourth, we verify the advantage of GigaGAN’s upsampler over other upsamplers in both conditional and unconditional super-resolution tasks. Lastly, we show our large-scale GANs still enjoy the continuous and disentangled latent space manipulation of GANs, enabling new image editing modes (Section 4.6).

4.1. Training and evaluation details

We implement GigaGAN based on the StudioGAN PyTorch library [37], following the standard FID evaluation protocol with the anti-aliasing bicubic resize function [67], unless otherwise noted. For text-to-image synthesis, we train our models on the union of LAION2B-en [88] and COYO-700M [8] datasets, with the exception of the 128-to-1024 upsampler model trained on Adobe’s internal Stock images. The image-text pairs are preprocessed based on CLIP score [24], image resolution, and aesthetic score [87], similar to prior work [78]. We use CLIP ViT-L/14 [71] for the pre-trained text encoder and OpenCLIP ViT-G/14 [32]
Figure 6. **Style mixing.** Our GAN-based architecture retains a disentangled latent space, enabling us to blend the coarse style of one sample with the fine style of another. All outputs are generated with the prompt "A Toy sport sedan, CG art." The corresponding latent codes are spliced together to produce a style-swapping grid.

Figure 7. **Prompt interpolation.** GigaGAN enables smooth interpolation between prompts, as shown in the interpolation grid. The four corners are generated from the same latent \( z \) but with different text prompts. The corresponding text embeddings \( t \) and style vectors \( w \) are interpolated to create a smooth transition. The same \( z \) results in similar layouts. See Figure 8 for more precise control.
Figure 8. **Prompt mixing.** GigaGAN retains a disentangled latent space, enabling us to combine the coarse style of one sample with the fine style of another. Moreover, GigaGAN can directly control the style with text prompts. Here we generate four outputs using the prompts “a X on tabletop”, shown in the “no mixing” column. Then we re-compute the text embeddings $t$ and the style codes $w$ using the new prompts “a X with the texture of Y on tabletop”, such as “a cube with the texture of crochet on tabletop”, and apply them to the second half layers of the generator, achieving layout-preserving fine style control. Cross-attention mechanism automatically localizes the style to the object of interest.

4.2. Effectiveness of proposed components

First, we show the effectiveness of our formulation via ablation study in Table 1. We set up a baseline by adding text-conditioning to StyleGAN2 and tuning the configuration based on the findings of StyleGAN-XL. We first directly increase the model size of this baseline, but we find that this does not improve the FID and CLIP scores. Then, we add our components one by one and observe that they consistently improve performance. In particular, our model is more scalable, as the higher-capacity version of the final formulation achieves better performance.

4.3. Text-to-Image synthesis

We proceed to train a larger model by increasing the capacity of the base generator and upsampler to 652.5M and 359.1M, respectively. This results in an unprecedented size of GAN model, with a total parameter count of 1.0B. Table 2 compares the performance of our end-to-end pipeline to various text-to-image generative models [5, 10, 63, 74, 75, 78–80, 101, 108]. Note that there exist differences in the training dataset, the pretrained text encoders, and even image resolutions. For example, GigaGAN initially synthesizes 512px images, which are resized to 256px before evaluation.

Table 2 shows that GigaGAN exhibits a lower FID than DALL·E 2 [74], Stable Diffusion [78], and Parti-750M [101]. While our model can be optimized to better match the feature distribution of real images than existing
models, the quality of the generated images is not necessarily better (see Appendix C for more samples). We acknowledge that this may represent a corner case of zero-shot FID where images are resized to 512px. However, when using unconditional Real-ESRGAN [33] and Stable Diffusion Upscaler [78], GigaGAN achieves a lower FID than the distilled Stable Diffusion while showing better performance speed compared to the progressively distilled diffusion models [59] for fast inference. As GigaGAN generates outputs in a single feedforward pass, the inference speed is still faster. The evaluation setup is different from Table 2 to match SD-distilled’s protocol [59].

Table 5. Unconditional 64→256 super-resolution on ImageNet.

<table>
<thead>
<tr>
<th>Model</th>
<th># Images</th>
<th># Steps</th>
<th>IS ↑</th>
<th>FID-5k ↓</th>
<th>PSNR ↑</th>
<th>SSIM ↑</th>
</tr>
</thead>
<tbody>
<tr>
<td>U-Net regression [81]</td>
<td>625M</td>
<td>5</td>
<td>121.1</td>
<td>0.30</td>
<td>27.9</td>
<td>0.80</td>
</tr>
<tr>
<td>SD [81]</td>
<td>625M</td>
<td>100</td>
<td>180.1</td>
<td>0.30</td>
<td>26.4</td>
<td>0.75</td>
</tr>
<tr>
<td>LDM-4 [79]</td>
<td>169M</td>
<td>100</td>
<td>166.3</td>
<td>0.37</td>
<td>24.4</td>
<td>0.69</td>
</tr>
<tr>
<td>emplLDM-4 [79]</td>
<td>552M</td>
<td>100</td>
<td>174.9</td>
<td>0.30</td>
<td>24.7</td>
<td>0.71</td>
</tr>
<tr>
<td>LDM-4-G [79]</td>
<td>183M</td>
<td>100</td>
<td>153.7</td>
<td>0.30</td>
<td>25.8</td>
<td>0.70</td>
</tr>
<tr>
<td>GigaGAN</td>
<td>693M</td>
<td>1</td>
<td>191.5</td>
<td>1.2</td>
<td>24.3</td>
<td>0.71</td>
</tr>
</tbody>
</table>

The inference speed of diffusion models. We compare GigaGAN with progressively distilled Stable Diffusion (SD-distilled) [59]. Table 3 demonstrates that GigaGAN remains faster than the distilled Stable Diffusion while showing better FID and CLIP scores of 21.1 and 0.32, respectively. We follow the evaluation protocol of SD-distilled [59] and report FID and CLIP scores on COCO2017 dataset [54], where images are resized to 512px.

4.5. Super-resolution for large-scale image synthesis

We separately evaluate the performance of the GigaGAN upsampler. Our evaluation consists of two parts. First, we compare GigaGAN with several commonly-used upsamplers. For the text-conditioned upsampling task, we combine the Stable Diffusion [78] 4x Upscaler and 2x Latent Upscaler to establish an 8x upsampling model (SD Upscaler). We also use the unconditional Real-ESRGAN [33] as another baseline. Table 4 measures the performance of the upsampler on random 10K images from the LAION dataset and shows that our GigaGAN upsampler signifi-
and style code

t text inputs. In detail, we can compute the text embedding
the disentangled style manipulation can be controlled via
potential for image synthesis. In Figure 8, we show that
bedding

t= \[ W \]

more, our model possesses another latent space of text em-
techniques of StyleGAN can transfer to GigaGAN. Further-
tangled

W

GAN, Figure 6 illustrates that GigaGAN maintains a disen-
twise, we perform coarse and fine-grained style swapping
using style vectors
w

using different prompts and apply them
to different layers of the generator. This way, we gain not
only the coarse and fine style disentanglement but also an
intuitive prompt-based maneuver in the style space.

5. Discussion and Limitations

Our experiments provide a conclusive answer about the
scalability of GANs: our new architecture can scale up to
model sizes that enable text-to-image synthesis. However,
the visual quality of our results is not yet comparable to
production-grade models like DALL-E 2. Figure 9 shows
several instances where our method fails to produce high-
quality results when compared to DALL-E 2, in terms of
photorealism and text-to-image alignment for the same in-
put prompts used in their paper.

Nevertheless, we have tested capacities well beyond
what is possible with a naïve approach and achieved com-
petitive results with autoregressive and diffusion models
trained with similar resources while being orders of magni-
tude faster and enabling latent interpolation and stylization.
Our GigaGAN architecture opens up a whole new design
space for large-scale generative models and brings back key
editing capabilities that became challenging with the tran-
sition to autoregressive and diffusion models. We expect
our performance to improve with larger models, as seen in
Table 1.

Acknowledgments. We thank Simon Niklaus, Alexandru
Chiculita, and Markus Woodson for building the distributed
training pipeline. We thank Nupur Kumari, Gaurav Parmar,
Bill Peebles, Phillip Isola, Alyosha Efros, and Joonghyuk
Shin for their helpful comments. We also want to thank
Chenlin Meng, Chitwan Saharia, and Jiahui Yu for answer-
ing many questions about their fantastic work. We thank
Kevin Duarte for discussions regarding upsampling beyond
4K. Part of this work was done while Minguk Kang was an
intern at Adobe Research. Minguk Kang and Jaesik Park
were supported by IITP grant funded by the government of
South Korea (MSIT) (POSTECH GSAI: 2019-0-01906 and
Image restoration: 2021-0-00537).

References

[1] Rameen Abdal, Yipeng Qin, and Peter Wonka. Im-
age2stylegan: How to embed images into the stylegan la-
tent space? In IEEE International Conference on Computer
Vision (ICCV), 2019. 5

super-resolution. IEEE Transactions on Pattern Analysis
and Machine Intelligence (TPAMI), 2020. 5

Wasserstein Generative Adversarial Networks. In Inter-
national Conference on Machine Learning (ICML), 2017. 7

Neural machine translation by jointly learning to align and
translate. In International Conference on Learning Repre-
sentations (ICLR), 2015. 7

[5] Yogesh Balaji, Seunjun Nah, Xun Huang, Arash Vahdat,
Jaiming Song, Karsten Kreis, Miika Aittala, Timo Aila,
Samuli Laine, Bryan Catanzaro, et al. ediffi: Text-to-
image diffusion models with an ensemble of expert denoi-

Scale GAN Training for High Fidelity Natural Image Syn-
thesis. In International Conference on Learning Repre-
sentations (ICLR), 2019. 1, 5, 6, 7, 18, 22

Figure 9. Failure cases. Our outputs with the same prompts as
DALL-E 2. Each column conditions on “a teddy bear on a skate-
board in Times Square”, “a Vibrant portrait painting of Salvador
Dali with a robotic half face”, and “A close up of a handpalm with
leaves growing from it”. Compared to production-grade models
such as DALL-E 2, our model exhibits limitations in realism and
compositionality. See Appendix C for uncurated comparisons.

cantly outperforms the other upsamplers in realism scores
(FID and patch-FID [9]), text alignment (CLIP score) and
closeness to the ground truth (LPIPS [105]). In addition,
for more controlled comparison, we train our model on
the ImageNet unconditional superresolution task and com-
pare performance with the diffusion-based models, includ-
ing SR3 [81] and LDM [79]. As shown in Table 5, Gi-
gaGAN achieves the best IS and FID scores with a single
feedforward pass.

4.6. Controllable image synthesis

StyleGANs are known to possess a linear latent space
useful for image manipulation, called the \( \mathcal{W} \)-space. Like-
wise, we perform coarse and fine-grained style swapping
using style vectors \( w \). Similar to the \( \mathcal{W} \)-space of Style-
GAN, Figure 6 illustrates that GigaGAN maintains a disen-
tangled \( \mathcal{W} \)-space, suggesting existing latent manipulation
techniques of StyleGAN can transfer to GigaGAN. Further-
more, our model possesses another latent space of text em-
bedding \( t = [ t_{\text{local}}, t_{\text{global}} ] \) prior to \( \mathcal{W} \), and we explore its
potential for image synthesis. In Figure 8, we show that
the disentangled style manipulation can be controlled via
text inputs. In detail, we can compute the text embedding
\( t \) and style code \( w \) using different prompts and apply them
to different layers of the generator. This way, we gain not
only the coarse and fine style disentanglement but also an
intuitive prompt-based maneuver in the style space.


[76] Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. In International Conference on Machine Learning (ICML), 2016. 5, 8


[87] Christoph Schuhmann. CLIP+MLP Aesthetic Score Predictor. https://github.com/christophschuhmann/improved-aesthetic-predictor. 9, 18

[88] Christoph Schuhmann. CLIP+MLP Aesthetic Score Predictor. https://github.com/christophschuhmann/improved-aesthetic-predictor. 9, 18


[99] Tao Xu, Pengchuan Zhang, Qiuyuan Huang, Han Zhang, Zhe Gan, Xiaolei Huang, and Xiaodong He. Attnagan: Fine-grained text to image generation with attentional generative adversarial networks. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018. 5


[104] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris N Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In IEEE International Conference on Computer Vision (ICCV), 2017. 1, 5, 8


Appendices

We first provide training and evaluation details in Appendix A. Then, we share results on ImageNet, with visual comparison to existing methods in Appendix B. Lastly in Appendix C, we show more visuals on our text-to-image synthesis results and compare them with LDM [79], Stable Diffusion [78], and DALL·E 2 [74].

A. Training and evaluation details

A.1. Text-to-image synthesis

We train GigaGAN on a combined dataset of LAION2B-en [88] and COYO-700M [5] in PyTorch framework [68]. For training, we apply center cropping, which results in a square image whose length is the same as the shorter side of the original image. Then, we resize the image to the resolution $64 \times 64$ using PIL.LANCZOS [94] resizer, which supports anti-aliasing [67]. We filter the training images based on image resolution ($\geq 512$), CLIP score ($>0.3$) [24], aesthetics score ($>5.0$) [87], and remove watermarked images. We train our GigaGAN based on the configurations denoted in the fourth and fifth columns of Table A2.

For evaluation, we use 40,504 and 30,000 real and generated images from COCO2014 [54] validation dataset as described in Imagen [80]. We apply the center cropping and resize the real and generated images to $299 \times 299$ resolution using PIL.BICUBIC, suggested by clean-fid [67]. We use the clean-fid library [67] for FID calculation.

A.2. Conditional image synthesis on ImageNet

We follow the training and evaluation protocol proposed by Kang et al. [37] to make a fair comparison against other cutting-edge generative models. We use the same cropping strategy to process images for training and evaluation as in our text-to-image experiments. Then, we resize the image to the target resolution ($64 \times 64$ for the base generator or $256 \times 256$ for the super-resolution stack) using PIL.LANCZOS [94] resizer, which supports anti-aliasing [67]. Using the pre-processed training images, we train GigaGAN based on the configurations denoted in the second and third columns of Table A2.

For evaluation, we upsample the real and generated images to $299 \times 299$ resolution using the PIL.BICUBIC resizer. To compute FID, we generate 50k images without truncation tricks [6, 41] and compare those images with the entire training dataset. We use the pre-calculated features of real images provided by StudioGAN [37] and 50k generated images for Precision & Recall [46] calculation.

A.3. Super-resolution results

For model training, we preprocess ImageNet in the same way as in Section A.2 and use the configuration in the last column of Table A2. To compare our model with SR3 [81] and LDM fairly, we follow the evaluation procedure described in SR3 and LDM papers.

B. ImageNet experiments

B.1. Qualitative results

We train a class-conditional GAN on the ImageNet dataset [13], for which apples-to-apples comparison is possible using the same dataset and evaluation pipeline. Our GAN achieves comparable generation quality to the cutting-edge generative models without a pretrained ImageNet classifier, which acts favorably toward automated metrics [45]. We apply L2 self-attention, style-adaptive convolution kernel, and matching-aware loss to our model and use a wider synthesis network to train the base 64px model with a batch size of 1024. Additionally, we train a separate 256px class-conditional upsampler model and combine them with an end-to-end finetuning stage. Table A1 shows that our method generates high-fidelity images.

Table A1. Class-conditional synthesis on ImageNet 256px. Our method performs competitively against large diffusion and transformer models. Shaded methods leverage a pretrained ImageNet classifier at training or inference time, which could act favorably toward the automated metrics [45]. † indicates IS [82] and FID [25] are borrowed from the original DiT paper [70].

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ADM-G-U</td>
<td>207.86</td>
<td>4.48</td>
<td>0.84/0.62</td>
<td>608M</td>
</tr>
<tr>
<td>ADM-G-U</td>
<td>240.24</td>
<td>4.01</td>
<td>0.85/0.62</td>
<td>726M</td>
</tr>
<tr>
<td>CDM [27]</td>
<td>158.71</td>
<td>4.88</td>
<td>-/-</td>
<td></td>
</tr>
<tr>
<td>LDM-8-G [79]</td>
<td>209.52</td>
<td>7.76</td>
<td>-/-</td>
<td>506M</td>
</tr>
<tr>
<td>LDM-8-G [79]</td>
<td>247.67</td>
<td>3.60</td>
<td>-/-</td>
<td>400M</td>
</tr>
<tr>
<td>DiT-XL/2† [70]</td>
<td>273.24</td>
<td>2.27</td>
<td>-/-</td>
<td>675M</td>
</tr>
<tr>
<td>Mask-GIT [11]</td>
<td>216.38</td>
<td>5.40</td>
<td>0.87/0.60</td>
<td>227M</td>
</tr>
<tr>
<td>VQ-GAN [19]</td>
<td>314.61</td>
<td>5.20</td>
<td>0.81/0.57</td>
<td>14B</td>
</tr>
<tr>
<td>RQ-Transformer [48]</td>
<td>339.41</td>
<td>3.83</td>
<td>0.85/0.60</td>
<td>3.8B</td>
</tr>
<tr>
<td>GigaGAN</td>
<td>225.52</td>
<td>3.45</td>
<td>0.84/0.61</td>
<td>569M</td>
</tr>
</tbody>
</table>

B.2. Quantitative results

We provide visual results from ADM-G-U, LDM, StyleGAN-XL [86], and GigaGAN in Figures A1 and A2. Although StyleGAN-XL has the lowest FID, its visual quality appears worse than ADM and GigaGAN. StyleGAN-XL struggles to synthesize the overall image structure, leading to less realistic images. In contrast, GigaGAN appears to synthesize the overall structure better than StyleGAN-XL and faithfully captures fine-grained details, such as the wing patterns of a monarch and the white fur of an arctic fox. Compared to GigaGAN, ADM-G-U synthesizes the image structure more rationally but lacks in reflecting the aforementioned fine-grained details.
Table A2. **Hyperparameters for GigaGAN training.** We denote Projection Discriminator [62] as PD, R1 regularization [61] as R1, Learned Perceptual Image Patch Similarity [105] as LPIPS, Adam with decoupled weight decay [56] as AdamW, and the pretrained ViT-B/32 visual encoder [71] as CLIP-ViT-B/32-V.

<table>
<thead>
<tr>
<th>Task</th>
<th>Class-Label-to-Image</th>
<th>Text-to-Image</th>
<th>Super-Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Dataset &amp; Resolution</strong></td>
<td>ImageNet 64</td>
<td>ImageNet 64→256</td>
<td>LAION/COYO 64</td>
</tr>
<tr>
<td>z dimension</td>
<td>64</td>
<td>128</td>
<td>128</td>
</tr>
<tr>
<td>w dimension</td>
<td>512</td>
<td>512</td>
<td>1024</td>
</tr>
<tr>
<td>Adversarial loss type</td>
<td>Logistic</td>
<td>Logistic</td>
<td>Logistic</td>
</tr>
<tr>
<td>Conditioning loss type</td>
<td>PD</td>
<td>PD</td>
<td>MS-I/O</td>
</tr>
<tr>
<td>R1 strength</td>
<td>0.2048</td>
<td>0.2048</td>
<td>0.2048</td>
</tr>
<tr>
<td>R1 interval</td>
<td>16</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>G Matching loss strength</td>
<td>-</td>
<td>-</td>
<td>1.0</td>
</tr>
<tr>
<td>D Matching loss strength</td>
<td>-</td>
<td>-</td>
<td>1.0</td>
</tr>
<tr>
<td>LPIPS strength</td>
<td>-</td>
<td>100.0</td>
<td>10.0</td>
</tr>
<tr>
<td>Optimizer</td>
<td>AdamW</td>
<td>AdamW</td>
<td>AdamW</td>
</tr>
<tr>
<td>Batch size</td>
<td>1024</td>
<td>256</td>
<td>512→1024</td>
</tr>
<tr>
<td>G learning rate</td>
<td>0.0025</td>
<td>0.0025</td>
<td>0.0025</td>
</tr>
<tr>
<td>D learning rate</td>
<td>0.0025</td>
<td>0.0025</td>
<td>0.0025</td>
</tr>
<tr>
<td>β₁ for AdamW</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>β₂ for AdamW</td>
<td>0.99</td>
<td>0.99</td>
<td>0.99</td>
</tr>
<tr>
<td>Weight decay strength</td>
<td>0.00001</td>
<td>0.00001</td>
<td>0.00001</td>
</tr>
<tr>
<td>Weight decay strength on attention</td>
<td>-</td>
<td>0.01</td>
<td>0.01</td>
</tr>
<tr>
<td># D updates per G update</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>G ema beta</td>
<td>0.9651</td>
<td>0.9912</td>
<td>0.9999</td>
</tr>
<tr>
<td>Precision</td>
<td>TF32</td>
<td>TF32</td>
<td>TF32</td>
</tr>
<tr>
<td>Mapping Network M layer depth</td>
<td>2</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Text Transformer T layer depth</td>
<td>-</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>G channel base</td>
<td>32768</td>
<td>32768</td>
<td>16384</td>
</tr>
<tr>
<td>D channel base</td>
<td>32768</td>
<td>32768</td>
<td>16384</td>
</tr>
<tr>
<td>G channel max</td>
<td>512</td>
<td>512</td>
<td>1600</td>
</tr>
<tr>
<td>D channel max</td>
<td>768</td>
<td>512</td>
<td>1536</td>
</tr>
<tr>
<td>G # of filters N for adaptive kernel selection</td>
<td>8</td>
<td>4</td>
<td>[1, 1, 2, 4, 8]</td>
</tr>
<tr>
<td>Attention type</td>
<td>self</td>
<td>self</td>
<td>self + cross</td>
</tr>
<tr>
<td>Attention resolutions</td>
<td>[8, 16, 32]</td>
<td>[16, 32]</td>
<td>[16, 32]</td>
</tr>
<tr>
<td>D attention resolutions</td>
<td>[8, 16, 32]</td>
<td>-</td>
<td>[8, 16, 32]</td>
</tr>
<tr>
<td>G attention depth</td>
<td>[4, 4, 4]</td>
<td>[2, 4, 4]</td>
<td>[2, 2, 1]</td>
</tr>
<tr>
<td>D attention depth</td>
<td>[1, 1, 1]</td>
<td>-</td>
<td>[2, 2, 1]</td>
</tr>
<tr>
<td>Attention dimension multiplier</td>
<td>1.0</td>
<td>1.4</td>
<td>1.0</td>
</tr>
<tr>
<td>MLP dimension multiplier of attention</td>
<td>4.0</td>
<td>4.0</td>
<td>4.0</td>
</tr>
<tr>
<td># synthesis block per resolution</td>
<td>1</td>
<td>5</td>
<td>[3, 3, 3, 2, 2]</td>
</tr>
<tr>
<td># discriminator block per resolution</td>
<td>1</td>
<td>1</td>
<td>[1, 2, 2, 2, 2]</td>
</tr>
<tr>
<td>Residual gain</td>
<td>1.0</td>
<td>0.4</td>
<td>0.4</td>
</tr>
<tr>
<td>Residual gain on attention</td>
<td>1.0</td>
<td>0.3</td>
<td>0.3</td>
</tr>
<tr>
<td>MinibatchStdLayer</td>
<td>True</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>D epilogue minibatch group size</td>
<td>8</td>
<td>4</td>
<td>True</td>
</tr>
<tr>
<td>Multi-scale training</td>
<td>False</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>Multi-scale loss ratio (high to low res)</td>
<td>-</td>
<td>-</td>
<td>[0.33, 0.17, 0.17, 0.17, 0.17]</td>
</tr>
<tr>
<td>D intermediate layer adv loss weight</td>
<td>-</td>
<td>-</td>
<td>0.01</td>
</tr>
<tr>
<td>D intermediate layer matching loss weight</td>
<td>-</td>
<td>-</td>
<td>0.05</td>
</tr>
<tr>
<td>Vision-aided discriminator backbone</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>G Model size</td>
<td>209.5M</td>
<td>359.8M</td>
<td>652.5M</td>
</tr>
<tr>
<td>D Model size</td>
<td>76.7M</td>
<td>30.7M</td>
<td>381.4M</td>
</tr>
<tr>
<td>Iterations</td>
<td>300k</td>
<td>620k</td>
<td>1350k</td>
</tr>
<tr>
<td># A100 GPUs for training</td>
<td>64</td>
<td>64</td>
<td>96–128</td>
</tr>
</tbody>
</table>

G Model size 209.5M, 359.8M, 652.5M, 359.1M, 359.0M
D Model size 76.7M, 30.7M, 381.4M, 130.1M, 28.9M
Iterations 300k, 620k, 1350k, 915k, 160k
# A100 GPUs for training 64, 64, 96–128, 64, 32
Figure A1. Uncurated images (above: Tench and below: Monarch) from ADM-G-U [15], LDM-4-G [79], GigaGAN (ours), and StyleGAN-XL [86]. FID values of each generative model are 4.01, 3.60, 3.45, and 2.32, respectively.
Figure A2. Uncurated images (above: Lorikeet and below: Arctic fox) from ADM-G-U [15], LDM-4-G [79], GigaGAN (ours), and StyleGAN-XL [86]. FID values of each generative model are 4.01, 3.60, 3.45, and 2.32, respectively.
C. Text-to-image synthesis results

C.1. Truncation trick at inference

Similar to the classifier guidance [15] and classifier-free guidance [28] used in diffusion models such as LDM, our GAN model can leverage the truncation trick [6, 41] at inference time.

\[ w_{\text{trunc}} = \text{lerp}(w_{\text{mean}}, w, \psi), \]  

(9)

where \( w_{\text{mean}} \) is the mean of \( w \) of the entire dataset, which can be precomputed. In essence, the truncation trick lets us trade diversity for fidelity by interpolating the latent vector to the mean of the distribution and thereby making the outputs more typical. When \( \psi = 1.0 \), \( w_{\text{mean}} \) is not used, and there is no truncation. When \( \psi = 0.0 \), \( w \) collapses to the mean, losing diversity.

While it is straightforward to apply the truncation trick for the unconditional case, it is less clear how to achieve this for text-conditional image generation. We find that interpolating the latent vector toward both the mean of the entire distribution as well as the mean of \( w \) conditioned on the text prompt produces desirable results.

\[ w_{\text{trunc}} = \text{lerp}(w_{\text{mean,c}}, \text{lerp}(w_{\text{mean}}, w, \psi), \psi), \]  

(10)

where \( w_{\text{mean,c}} \) can be computed at inference time by sampling \( w = M(z, c) \) 16 times with the same \( c \), and taking the average. This operation’s overhead is negligible, as the mapping network \( M \) is computationally light compared to the synthesis network. At \( \psi = 1.0 \), \( w_{\text{trunc}} \) becomes \( w_{\text{trunc}} = w \), meaning no truncation. Figure A4 demonstrates the effect of our text-conditioned truncation trick.

Quantitatively, the effect of truncation is similar to the guidance technique of diffusion models. As shown in Figure A3, the CLIP score increases with more truncation, where the FID increases due to reduced diversity.

C.2. Comparison to diffusion models

Finally, we show randomly sampled results of our model and compare them with publicly available diffusion models, LDM [79], Stable Diffusion [78], and DALL·E 2 [74].

Figure A3. We investigate how our FID and CLIP score changes over different truncation values \([1.0, 0.9, 0.8, 0.7, 0.6, 0.5]\), by visualizing them along with the FID-CLIP score curve of two publicly available large scale diffusion models: LDM and Stable Diffusion. It is seen that the CLIP score increases with more truncation, at the cost of reduced diversity indicated by higher FID. The guidance values of the diffusion models are \([1.0, 1.25, 1.5, 1.75, 2, 4, 6, 8, 10]\).
Figure A4. The visual effect of our truncation trick. We demonstrate the effect of truncation by decreasing the truncation value $\psi$ from 1.0. We show six example outputs with the text prompt “digital painting of a confident and severe looking northern war goddess, extremely long blond braided hair, beautiful blue eyes and red lips.” and “Magritte painting of a clock on a beach.”. At 1.0 (no truncation), the diversity is high, but the alignment is not satisfactory. As the truncation increases, text-image alignment improves, at the cost of diversity. We find that a truncation value between 0.8 and 0.7 produces the best result.
Figure A5. **Style mixing.** GigaGAN maintains a disentangled latent space, allowing us to blend the coarse style of one sample with the fine style of another. The corresponding latent codes are spliced together to produce a style-swapping grid. The outputs are generated from the same prompt but with different latent codes.
Figure A6. **Prompt interpolation.** GigaGAN enables smooth interpolation between prompts, as shown in the interpolation grid. The four corners are generated from the same latent but with different text prompts. The corresponding text embeddings and style vectors are interpolated to create a smooth transition. The same results in similar layouts.
<table>
<thead>
<tr>
<th>“Photography”</th>
<th>“Pixel art”</th>
<th>“pencil drawing”</th>
<th>“Claude Monet painting”</th>
<th>“Van Gogh painting”</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure A7. **Prompt mixing.** GigaGAN can directly control the style with text prompts. Here we generate five outputs using the prompts “Photography of X”, shown in the “Photography” column. Then we re-compute the text embeddings $t$ and the style codes $w$ using the new prompts “Y of X”, such as “Van Gogh painting of the black hole in the space”, and apply them to the second half layers of the generator, achieving layout-preserving style control. Cross-attention mechanism automatically localizes the style to the object of interest. We use the following prompts in order from the row above. (1) the black hole in the space. (2) a teacup on the desk. (3) a table top with a vase of flowers on it. (4) a birthday cake. (5) a beautiful flower. We discover that GigaGAN’s prompt-based style transfer is only possible for images of a single and simple object.
Figure A8. **Diversity of synthesized images using GigaGAN.** GigaGAN can synthesize diverse images for a given prompt. We use the following prompts in order from the row above. (1) Majestic castle and fireworks, artwork, oil painting, 2k. (2) Oil-painting depicting a sunset over the sea with waves. (3) A burning candle with two wicks, detailed photo, studio lighting. (4) Portrait of Isaac Newton, long hair. (5) An abstract representation of the speed of light.
“A loft bed with a dresser underneath it.”

Figure A9. Random outputs of our model, Latent Diffusion Model [79], Stable Diffusion [78], and DALL·E 2 [74], using prompt “A loft bed with a dresser underneath it”. We show two versions of our model, one without truncation and the other with truncation. Our model enjoys faster speed than the diffusion models. Still, we observe our model falls behind in structural coherency, such as the number of legs of the bed frames. For LDM and Stable Diffusion, we use 250 and 50 sampling steps with DDIM / PLMS [55], respectively. For DALL·E 2, we generate images using the official DALL·E service [64].
“A green vase filed with red roses sitting on top of table.”

Figure A10. Random outputs of our model, Latent Diffusion Model [79], Stable Diffusion [78], and DALL-E 2 [74], using prompt “A green vase filed with red roses sitting on top of table”. We show two versions of our model, one without truncation and the other with truncation. Our model enjoys faster speed than the diffusion models in both cases. Still, we observe our model falls behind in structural coherency like the symmetry of the vases. For LDM and Stable Diffusion, we use 250 and 50 sampling steps with DDIM / PLMS [55], respectively. For DALL-E 2, we generate images using the official DALL-E service [64].
Figure A11. Random outputs of our model, Latent Diffusion Model [79], Stable Diffusion [78], and DALL-E 2 [74], using prompt “A zebra in the grass who is cleaning himself”. We show two versions of our model, one without truncation and the other with truncation. Our model enjoys faster speed than the diffusion models in both cases. Still, we observe our model falls behind in details, such as the precise stripe pattern of the positioning of eyes. For LDM and Stable Diffusion, we use 250 and 50 sampling steps with DDIM / PLMS [55], respectively. For DALL-E 2, we generate images using the official DALL-E service [64].
“A teddy bear on a skateboard in times square.”

Figure A12. Random outputs of our model, Latent Diffusion Model [79], Stable Diffusion [78], and DALL·E 2 [74], using prompt “A teddy bear on a skateboard in times square”. We show two versions of our model, one without truncation and the other with truncation. Our model enjoys faster speed than the diffusion models in both cases. Still, we observe our model falls behind in details, like the exact shape of skateboards. For LDM and Stable Diffusion, we use 250 and 50 sampling steps with DDIM / PLMS [55], respectively. For DALL·E 2, we generate images using the official DALL·E service [64].
“Vibrant portrait painting of Salvador Dalí with a robotic half face.”

Figure A13. Random outputs of our model, Latent Diffusion Model [79], Stable Diffusion [78], and DALL·E 2 [74], using prompt “Vibrant portrait painting of Salvador Dalí with a robotic half face”. We show two versions of our model, one without truncation and the other with truncation. Our model enjoys faster speed than the diffusion models in both cases. Still, we observe our model falls behind in structural details like in the detailed shape of eyes. For LDM and Stable Diffusion, we use 250 and 50 sampling steps with DDIM / PLMS [55], respectively. For DALL·E 2, we generate images using the official DALL·E service [64].
“Three men in military suits are sitting on a bench.”

Figure A14. Random outputs of our model, Latent Diffusion Model [79], Stable Diffusion [78], and DALL·E 2 [74], using prompt “Three men in military suits are sitting on a bench”. We show two versions of our model, one without truncation and the other with truncation. Our model enjoys faster speed than the diffusion models in both cases. Still, we observe our model falls behind in details in facial expression and attire. For LDM and Stable Diffusion, we use 250 and 50 sampling steps with DDIM / PLMS [55], respectively. For DALL·E 2, we generate images using the official DALL·E service [64].
Figure A15. **Our GAN-based upsampler** can serve as the upsampler for many text-to-image models that generate initial outputs at low resolutions like 64px or 128px. We simulate such usage by applying our 8× superresolution model on a low-res 128px artwork to obtain the 1K output, using “Portrait of a kitten dressed in a bow tie. Red Rose. Valentine’s day.”. Then our model can be re-applied to go beyond 4K. We compare our model with the text-conditioned upscaler of Stable Diffusion [78] and unconditional Real-ESRGAN [33]. Zooming in is recommended for comparison between 1K and 4K outputs.
Figure A16. **Our GAN-based upsampler** can serve as the upsampler for many text-to-image models that generate initial outputs at low resolutions like 64px or 128px. We simulate such usage by applying our 8× superresolution model on a low-res 128px artwork to obtain the 1K output, using “Heart shaped pancakes with honey and strawberry for Valentine’s Day”. Then our model can be re-applied to go beyond 4K. We compare our model with the text-conditioned upscaler of Stable Diffusion [78] and unconditional Real-ESRGAN [33]. Zooming in is recommended for comparison between 1K and 4K outputs.
Figure A17. Our GAN-based upsampler can also be used as an off-the-shelf superresolution model for real images with a large scaling factor by providing an appropriate description of the image. We apply our text-conditioned $8 \times$ superresolution model on a low-res 128px photo to obtain the 1K output, using “An elephant spraying water with its trunk”. Then our model can be re-applied to go beyond 4K. We compare our model with the text-conditioned upscaler of Stable Diffusion [78] and unconditional Real-ESRGAN [33]. Zooming in is recommended for comparison between 1K and 4K outputs.