The rise (and fall?) of GANs

M. Kang et al. Scaling up GANs for Text-to-Image Synthesis. CVPR 2023
The rise of GANs

4.5 years of GAN progress on face generation.

arxiv.org/abs/1406.2661 arxiv.org/abs/1511.06434
arxiv.org/abs/1606.07536 arxiv.org/abs/1710.10196
arxiv.org/abs/1812.04948
The rise of GANs

**EBGAN** (2017)

**BigGAN** (2018)
The golden age of GANs: The StyleGAN family

- 2018: Progressive GAN
- 2019: StyleGAN
- 2020: StyleGAN2
- 2021: StyleGAN3
- 2022: StyleGAN-XL
- 2023: GigaGAN
Recall: DCGAN architecture

Progressive GANs

Realistic face images up to 1024 x 1024 resolution

Progressive GANs

- Key idea: train lower-resolution models, gradually add layers corresponding to higher-resolution outputs

Progressive GANs

- Key idea: train lower-resolution models, gradually add layers corresponding to higher-resolution outputs

Transition from 16x16 to 32x32 images

Progressive GANs: Implementation details

- Loss: WGAN-GP loss (preferred) or LSGAN
- Architectures:
  - Nearest neighbor upsampling (2x2 replication) followed by regular convolutions instead of transposed conv layers
  - Average pooling instead of striding for downsampling in discriminator
  - Leaky ReLUs used in discriminator and generator
  - Per-pixel response normalization in generator: rescale feature vector in each pixel to unit length after each conv layer
- Use of minibatch standard deviation in discriminator (append to feature map)
- Exponential moving average of generator weights for display

Progressive GANs: Results

256 x 256 results for LSUN categories
StyleGAN

- Built on top of Progressive GAN

StyleGAN

- Built on top of Progressive GAN
- Start generation with constant (instead of noise vector)
- Noise vector is transformed to latent vector $w$ that is later specialized to style codes
- Style codes control *adaptive instance normalization* (AdaIN) or scaling and biasing of each feature map
- Add noise after each convolution and before nonlinearity (enables stochastic detail)

StyleGAN: Results

Mixing styles

“Two sets of images were generated from their respective latent codes (sources A and B); the rest of the images were generated by copying a specified subset of styles from source B and taking the rest from source A.”
Mixing styles
StyleGAN: Bedrooms

StyleGAN: Cars

StyleGAN2

- Change normalization, remove progressive growing to address StyleGAN artifacts

Figure 1. Instance normalization causes water droplet-like artifacts in StyleGAN images. These are not always obvious in the generated images, but if we look at the activations inside the generator network, the problem is always there, in all feature maps starting from the 64x64 resolution. It is a systemic problem that plagues all StyleGAN images.

Figure 6. Progressive growing leads to “phase” artifacts. In this example the teeth do not follow the pose but stay aligned to the camera, as indicated by the blue line.

T. Karras et al. Analyzing and Improving the Image Quality of StyleGAN. CVPR 2020
Figure 1: Examples of “texture sticking”. **Left:** The average of images generated from a small neighborhood around a central latent (top row). The intended result is uniformly blurry because all details should move together. However, with StyleGAN2 many details (e.g., fur) stick to the same pixel coordinates, showing unwanted sharpness. **Right:** From a latent space interpolation (top row), we extract a short vertical segment of pixels from each generated image and stack them horizontally (bottom). The desired result is hairs moving in animation, creating a time-varying field. With StyleGAN2 the hairs mostly stick to the same coordinates, creating horizontal streaks instead.

StyleGAN3

Random latent walk using directions from StyleCLIP, GANSpace, and SeFa.


Videos
Application of GANs: Inversion and image editing

W. Xia et al. GAN Inversion: A Survey, TPAMI 2023
GAN-based image editing

Figure 7. Manipulation of real images using encoder-based inversion. Original images are from FFHQ, and were not part of the encoder’s training set.

Z. Wu et al. StyleSpace Analysis: Disentangled Controls for StyleGAN Image Generation. CVPR 2021
Editing using text prompts

• Combine powerful recent image-text embedding technique (CLIP) with pre-trained StyleGAN for text-based image editing

O. Patashnik et al.. StyleCLIP: Text-Driven Manipulation of StyleGAN Imagery. ICCV 2021
The golden age of GANs: The StyleGAN family

• 2018: Progressive GAN
• 2019: StyleGAN
• 2020: StyleGAN2
• 2021: StyleGAN3
• In the meanwhile…
BigGAN

- Scale up self-attention GAN to generate ImageNet images up to 512 x 512 resolution

A. Brock, J. Donahue, K. Simonyan, *Large scale GAN training for high fidelity natural image synthesis*, ICLR 2019
BigGAN

The good

The bad
Diffusion models

A. Ramesh et al. Hierarchical text-conditional image generation with CLIP latents. arXiv 2022
The golden age of GANs: The StyleGAN family

- 2018: Progressive GAN
- 2019: StyleGAN
- 2020: StyleGAN2
- 2021: StyleGAN3
- 2022: StyleGAN-XL
- 2023: GigaGAN
Fig. 1. Class-conditional samples generated by StyleGAN3 (left) and StyleGAN-XL (right) trained on ImageNet at resolution 256$^2$.

A. Sauer et al. **StyleGAN-XL: Scaling StyleGAN to Large Diverse Datasets**, SIGGRAPH 2022
StyleGAN-XL: Details

- Generator based on StyleGAN3 with progressive growing and 3x more parameters
- Discriminator based on projected GANs
  - Recall the standard GAN objective:
    \[ V(G, D) = \mathbb{E}_{x \sim p_{data}} \log D(x) + \mathbb{E}_{z \sim p} \log(1 - D(G(z))) \]
  - Projected GAN objective introduces multiple discriminators operating on projections \( P_l \) from a fixed pre-trained feature space:
    \[ V(G, D) = \sum_l \left( \mathbb{E}_{x \sim p_{data}} \log D_l(P_l(x)) + \mathbb{E}_{z \sim p} \log(1 - D_l(P_l(G(z)))) \right) \]
  - Each \( P_l \) returns a feature map of a different resolution by applying random cross-channel mixing and cross-scale mixing to a pre-trained network

A. Sauer et al. Projected GANs Converge Faster. NeurIPS 2021
StyleGAN-XL: Details

Cross-channel mixing (CCM)

Cross-scale mixing (CSM)

Figure 2: CCM (dashed blue arrows) employs $1 \times 1$ convolutions with random weights.

Figure 3: CSM (dashed red arrows) adds random $3 \times 3$ convolutions and bilinear upsampling, yielding a U-Network.

A. Sauer et al. Projected GANs Converge Faster. NeurIPS 2021
StyleGAN-XL: Details

- Generator based on StyleGAN3 with progressive growing and 3x more parameters
- Discriminator based on projected GANs
- Both generator and discriminator are conditioned on pre-trained class embedding vectors
- **Classifier guidance**: add cross-entropy loss of pre-trained classifier to the generator objective to encourage output of classifier to have high probability for correct class
Fig. 2. **Training StyleGAN-XL.** We feed a latent code \( z \) and class label \( c \) to the pretrained embedding and the mapping network \( G_m \) to generate style codes \( w \). The codes modulate the convolutions of the synthesis network \( G_s \). During training, we gradually add layers to double the output resolution for each stage of the progressive growing schedule. We only train the latest layers while keeping the others fixed. \( G_m \) is only trained for the initial \( 16^2 \) stage and remains fixed for the higher-resolution stages. The synthesized image is upsampled when smaller than \( 224^2 \) and passed through a CNN and a ViT and respective feature mixing blocks (CCM+CSM). At higher resolutions, the CNN receives the unaltered image while the ViT receives a downsampled input to keep memory requirements low but still utilize its global feedback. Finally, we apply eight independent discriminators on the resulting multi-scale feature maps. The image is also fed to classifier CLF for classifier guidance.
# StyleGAN-XL: Results on ImageNet

<table>
<thead>
<tr>
<th>Model</th>
<th>Resolution 128^2</th>
<th>Resolution 256^2</th>
<th>Resolution 512^2</th>
<th>Resolution 1024^2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FID ↓  sFID ↓  rFID ↓  IS ↑  Pr ↑  Rec ↑</td>
<td>FID ↓  sFID ↓  rFID ↓  IS ↑  Pr ↑  Rec ↑</td>
<td>FID ↓  sFID ↓  rFID ↓  IS ↑  Pr ↑  Rec ↑</td>
<td>FID ↓  sFID ↓  rFID ↓  IS ↑  Pr ↑  Rec ↑</td>
</tr>
<tr>
<td>BigGAN</td>
<td>6.02  7.18  6.09  145.83  0.86  0.35</td>
<td>49.20</td>
<td>BigGAN</td>
<td>6.95  7.36  75.24  202.65  0.87  0.28</td>
</tr>
<tr>
<td>CDM</td>
<td>3.52  128.80  128.80</td>
<td>CDM</td>
<td>4.88  158.70  158.70</td>
<td>CDM</td>
</tr>
<tr>
<td>ADM</td>
<td>5.91  5.09  13.29  93.31  0.70  0.65</td>
<td>ADM</td>
<td>3.94  6.14  11.86  215.84  0.83  0.53</td>
<td></td>
</tr>
<tr>
<td>ADM-G</td>
<td>2.97  5.09  3.80  141.37  0.78  0.59</td>
<td>ADM-G</td>
<td>2.30  4.02  7.06  265.12  0.78  0.53</td>
<td></td>
</tr>
<tr>
<td>StyleGAN-XL</td>
<td><strong>1.81</strong>  <strong>3.82</strong>  <strong>1.82</strong>  <strong>200.55</strong>  <strong>0.77</strong>  <strong>0.55</strong></td>
<td>StyleGAN-XL</td>
<td><strong>2.30</strong>  <strong>4.02</strong>  <strong>7.06</strong>  <strong>265.12</strong>  <strong>0.78</strong>  <strong>0.53</strong></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model</th>
<th>Resolution 512^2</th>
<th>Resolution 1024^2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FID ↓  sFID ↓  rFID ↓  IS ↑  Pr ↑  Rec ↑</td>
<td>FID ↓  sFID ↓  rFID ↓  IS ↑  Pr ↑  Rec ↑</td>
</tr>
<tr>
<td>BigGAN</td>
<td>8.43  8.13  312.00  177.90  0.88  0.29</td>
<td><strong>2.52</strong>  <strong>4.12</strong>  <strong>413.12</strong>  <strong>260.14</strong>  <strong>0.76</strong>  <strong>0.51</strong></td>
</tr>
<tr>
<td>CDM</td>
<td>23.24  10.19  561.32  58.06  0.73  <strong>0.60</strong></td>
<td>CDM</td>
</tr>
<tr>
<td>ADM</td>
<td>3.85  5.86  210.83  221.72  0.84  0.53</td>
<td>ADM</td>
</tr>
<tr>
<td>ADM-G-U</td>
<td>2.41  4.06  51.54  267.75  0.77  0.52</td>
<td>ADM-G-U</td>
</tr>
<tr>
<td>StyleGAN-XL</td>
<td><strong>2.41</strong>  <strong>4.06</strong>  <strong>51.54</strong>  <strong>267.75</strong>  <strong>0.77</strong>  <strong>0.52</strong></td>
<td>StyleGAN-XL</td>
</tr>
</tbody>
</table>
Text-to-image synthesis: GigaGAN

M. Kang et al. Scaling up GANs for Text-to-Image Synthesis. CVPR 2023
GigaGAN: Generator

- Goal: catch up to diffusion models by training a GAN for text-to-image generation on 2B images from the LAION dataset
- 1B parameters, 6x larger than StyleGAN-XL, but smaller than diffusion models (Imagen: 3B, DALL-E 2: 5.5B, Parti: 20B)
- First generate at 64x64 resolution, then upsample to 512x512 using a separate network
- Architecture based on StyleGAN2, but with self-attention layers and sample-adaptive convolution kernel selection
GigaGAN: Generator

Figure 4. **Our GigaGAN high-capacity text-to-image generator.** First, we extract text embeddings using a pretrained CLIP model and a learned encoder $T$. The local text descriptors are fed to the generator using cross-attention. The global text descriptor, along with a latent code $z$, is fed to a style mapping network $M$ to produce style code $w$. The style code modulates the main generator using our style-adaptive kernel selection, shown on the right. The generator outputs an image pyramid by converting the intermediate features into RGB images. To achieve higher capacity, we use multiple attention and convolution layers at each scale (Appendix A2). We also use a separate upsampler model, which is not shown in this diagram.
Figure 5. **Our discriminator** consists of two branches for processing the image and the text conditioning $t_D$. The text branch processes the text similar to the generator (Figure 4). The image branch receives an image pyramid and makes independent predictions for each image scale. Moreover, the predictions are made at all subsequent scales of the downsampling layers, making it a *multi-scale input, multi-scale output* (MS-I/O) discriminator.
GigaGAN: Discriminator

- Multiple loss terms:
  - Standard adversarial loss (NSGAN)
  - Matching-aware loss where image is paired with random conditioning vector and discriminator is encouraged to label the pair as “fake”
  - CLIP contrastive loss: enforce high image-text similarity using pre-trained image and text encoders
  - “Vision-aided loss” similar to projected GAN
## GigaGAN: Ablation study

<table>
<thead>
<tr>
<th>Model</th>
<th>Image quality measure</th>
<th>Image-text coherence measure</th>
<th># Param.</th>
</tr>
</thead>
<tbody>
<tr>
<td>StyleGAN2</td>
<td>29.91</td>
<td>0.222</td>
<td>27.8M</td>
</tr>
<tr>
<td>+ Larger (5.7×)</td>
<td>34.07</td>
<td>0.223</td>
<td>158.9M</td>
</tr>
<tr>
<td>+ Tuned</td>
<td>28.11</td>
<td>0.228</td>
<td>26.2M</td>
</tr>
<tr>
<td>+ Attention</td>
<td>23.87</td>
<td>0.235</td>
<td>59.0M</td>
</tr>
<tr>
<td>+ Matching-aware D</td>
<td>27.29</td>
<td>0.250</td>
<td>59.0M</td>
</tr>
<tr>
<td>+ Matching-aware G and D</td>
<td>21.66</td>
<td>0.254</td>
<td>59.0M</td>
</tr>
<tr>
<td>+ Adaptive convolution</td>
<td>19.97</td>
<td>0.261</td>
<td>80.2M</td>
</tr>
<tr>
<td>+ Deeper</td>
<td>19.18</td>
<td>0.263</td>
<td>161.9M</td>
</tr>
<tr>
<td>+ CLIP loss</td>
<td>14.88</td>
<td>0.280</td>
<td>161.9M</td>
</tr>
<tr>
<td>+ Multi-scale training</td>
<td>14.92</td>
<td>0.300</td>
<td>164.0M</td>
</tr>
<tr>
<td>+ Vision-aided GAN</td>
<td>13.67</td>
<td>0.287</td>
<td>164.0M</td>
</tr>
<tr>
<td>+ Scale-up (GigaGAN)</td>
<td>9.18</td>
<td>0.307</td>
<td>652.5M</td>
</tr>
</tbody>
</table>
GigaGAN: Style mixing

“A Toy sport sedan, CG art.”

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.
GigaGAN: Prompt interpolation

“.. in a sunny day”

“A modern mansion ..”  “A victorian mansion ..”

“.. in sunset”

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.
GigaGAN: Prompt mixing

<table>
<thead>
<tr>
<th>Prompt</th>
<th>no mixing</th>
<th>“crochet”</th>
<th>“fur”</th>
<th>“denim”</th>
<th>“brick”</th>
</tr>
</thead>
<tbody>
<tr>
<td>“a cube on tabletop”</td>
<td><img src="image1.png" alt="Image" /></td>
<td><img src="image2.png" alt="Image" /></td>
<td><img src="image3.png" alt="Image" /></td>
<td><img src="image4.png" alt="Image" /></td>
<td><img src="image5.png" alt="Image" /></td>
</tr>
<tr>
<td>“a ball on tabletop”</td>
<td><img src="image6.png" alt="Image" /></td>
<td><img src="image7.png" alt="Image" /></td>
<td><img src="image8.png" alt="Image" /></td>
<td><img src="image9.png" alt="Image" /></td>
<td><img src="image10.png" alt="Image" /></td>
</tr>
<tr>
<td>“a teddy bear on tabletop”</td>
<td><img src="image11.png" alt="Image" /></td>
<td><img src="image12.png" alt="Image" /></td>
<td><img src="image13.png" alt="Image" /></td>
<td><img src="image14.png" alt="Image" /></td>
<td><img src="image15.png" alt="Image" /></td>
</tr>
<tr>
<td>“a teddy bear on tabletop”</td>
<td><img src="image16.png" alt="Image" /></td>
<td><img src="image17.png" alt="Image" /></td>
<td><img src="image18.png" alt="Image" /></td>
<td><img src="image19.png" alt="Image" /></td>
<td><img src="image20.png" alt="Image" /></td>
</tr>
</tbody>
</table>

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.
GigaGAN: Comparison with diffusion models

“A teddy bear on a skateboard in times square.”

Ours (512px, 0.13s / img)

Ours (512px, 0.14s / img, truncation $\psi = 0.8$)
GigaGAN: Comparison with diffusion models

Stable Diffusion v1.5 (512px, 2.9s / img, 50 steps, guidance=7.5)

DALL-E 2 (1024px)
GigaGAN: Comparison with diffusion models

“Vibrant portrait painting of Salvador Dalí with a robotic half face.”
GigaGAN: Comparison with diffusion models

Stable Diffusion v1.5 (512px, 2.9s / img, 50 steps, guidance=7.5)

DALL·E 2 (1024px)
GigaGAN: Quantitative evaluation

Table 2. **Comparison to recent text-to-image models.** Model size, GPU days, total images seen during training, COCO FID-30k, and inference speed of text-image models. * denotes that the model has been evaluated by us. GigaGAN achieves a lower FID than DALL·E 2 [74], Stable Diffusion [78], and Parti-750M [101], while being much faster compared to recent competitive methods.

<table>
<thead>
<tr>
<th>Model</th>
<th>Type</th>
<th># Param.</th>
<th># Images</th>
<th>FID-30k↓</th>
<th>Inf. time</th>
</tr>
</thead>
<tbody>
<tr>
<td>DALL·E [75]</td>
<td>Diff</td>
<td>12.0B</td>
<td>1.54B</td>
<td>27.50</td>
<td>-</td>
</tr>
<tr>
<td>GLIDE [63]</td>
<td>Diff</td>
<td>5.0B</td>
<td>5.94B</td>
<td>12.24</td>
<td>15.0s</td>
</tr>
<tr>
<td>LDM [79]</td>
<td>Diff</td>
<td>1.5B</td>
<td>0.27B</td>
<td>12.63</td>
<td>9.4s</td>
</tr>
<tr>
<td>DALL·E 2 [74]</td>
<td>Diff</td>
<td>5.5B</td>
<td>5.63B</td>
<td>10.39</td>
<td>-</td>
</tr>
<tr>
<td>Imagen [80]</td>
<td>Diff</td>
<td>3.0B</td>
<td>15.36B</td>
<td>7.27</td>
<td>9.1s</td>
</tr>
<tr>
<td>eDiff-I [5]</td>
<td>Diff</td>
<td>9.1B</td>
<td>11.47B</td>
<td>6.95</td>
<td>32.0s</td>
</tr>
<tr>
<td>Parti-750M [101]</td>
<td>AR</td>
<td>750M</td>
<td>3.69B</td>
<td>10.71</td>
<td>-</td>
</tr>
<tr>
<td>Parti-3B [101]</td>
<td>AR</td>
<td>3.0B</td>
<td>3.69B</td>
<td>8.10</td>
<td>6.4s</td>
</tr>
<tr>
<td>Parti-20B [101]</td>
<td>AR</td>
<td>20.0B</td>
<td>3.69B</td>
<td>7.23</td>
<td>-</td>
</tr>
<tr>
<td>LAFITE [108]</td>
<td>GAN</td>
<td>75M</td>
<td>-</td>
<td>26.94</td>
<td>0.02s</td>
</tr>
<tr>
<td>SD-v1.5* [78]</td>
<td>Diff</td>
<td>0.9B</td>
<td>3.16B</td>
<td>9.62</td>
<td>2.9s</td>
</tr>
<tr>
<td>Muse-3B [10]</td>
<td>AR</td>
<td>3.0B</td>
<td>0.51B</td>
<td>7.88</td>
<td>1.3s</td>
</tr>
<tr>
<td>GigaGAN</td>
<td>GAN</td>
<td>1.0B</td>
<td>0.98B</td>
<td>9.09</td>
<td>0.13s</td>
</tr>
</tbody>
</table>

“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... We acknowledge that this may represent a corner case of zero-shot FID on COCO2014 dataset and suggest that further research on a better evaluation metric is necessary to improve text-to-image models.”
Do GANs have a future?

• We have to stay tuned to find out…