DogeGAN-XL: end-to-end solution for generating NFTs

Alex Kashi
Harvard University
alexkashi@g.harvard.edu

Geoffrey Liu
Harvard University
geoffreyliu@fas.harvard.edu

Abstract

There has been a recent surge of data-centric approaches to computer graphics, using deep neural networks as generative models for photorealistic and artistic content creation. Recently, the introduction of Non-Fungible Tokens (NFTs) has created a marketplace where individual artists can submit their digital artwork. This market has seen a dramatic increase in demand, driving the prices of some digital artwork collections to millions of dollars. We propose an end-to-end solution to generating digital artwork, using Generative Adversarial Networks (GANs) to mimic popular collections of NFT artwork. The output of our GAN is then upsampled using a pre-trained super-resolution network. NFT artwork can also be generated to mimic the qualities of input images and text. Our end-to-end training pipeline, DogeGAN-XL, is designed to be computationally feasible, and our proposed pipeline can be trained in less than two days on a single NFT collection using a single GPU. There is a panoply of potential applications for our pipeline, from creating marketable NFTs to avatar generation in the metaverse.

1. Introduction

Generative Adversarial Networks (GANs) are a commonly used tool in computer vision for generating new content from examples. Typically two neural networks, a generator, and a discriminator, compete against each other. The generator's goal is to produce content new content from the distribution of the training data, while the discriminator judges if the generated image is a forgery or from the original distribution. GANs have become a popular neural architecture for generating content and has seen applications ranging from self-driving cars to the metaverse.

Non-fungible tokens (NFTs) are unique cryptographic tokens stored on a blockchain that can be sold or traded. NFTs have many use cases; however they are most commonly used for trading and verifying ownership over digital artwork. At the time of writing, the market value of NFTs is estimated to be $41B USD. Most of the highly-priced NFT artworks are bundled into collections of images ($\lesssim$10,000) with the same artistic style. The appeal of applying GANs to NFTs is obvious. GANs can generate digital content based on prior art worth a significant amount of crypto. However, whether the content GANs generate can become high-value is an open question. Many people have made this association and sold AI-generated NFTs on OpenSea, trading up to 194 Ethereum or USD 463,307 in volume [1][2].

Training GANs is notoriously tricky and computationally expensive. The training procedure is sensitive to the choice of neural network architecture, loss functions, and training hyperparameters like the learning rate of the discriminator and generator. The GAN training process is not guaranteed to converge. Most commonly available GAN implementations are trained using choices that perform well on natural image datasets, including ImageNet and high-quality FlickR Faces, rather than digital artwork. Typically, GANs require weeks to train, even on lower resolution images.

Our project proposes a computationally feasible pipeline for training GANs on NFT collections and explores and propose solutions to the unique peculiarities of training these models on NFT collections. Our final solution can be trained in $\lesssim$2 days using a single GPU and allows the generation of new, unique NFTs based on input images and text.

2. Related Work

Generative models that produce samples from high dimensional data distributions are widely used, especially in computer graphics. Many generative architectures are available to researchers, such as autoregressive models, variational autoencoders, and generative adversarial networks. Autoregressive models produce sharp, high-quality images. However, they are slow to train and do not have a latent representation. Variational Autoencoders are easier to train but tend to produce lower-quality images. GANs produce sharp

[1] https://opensea.io/collection/bastard-gan-punks
[2] https://opensea.io/collection/gan-apes-nft

images, although they are hard to scale to higher resolutions, have limited variation in the images they produce, and training tends to be unstable. Exhibit issues include mode collapse, vanishing gradients, and failure to converge. [11] Despite this, in 2022, GANs have become the predominant choice for image generation.

2.1. Generative Adversarial Networks

Generative Adversarial networks is a framework for estimating generative models via an adversarial process. Two networks are simultaneously trained: a generative model G that captures the data distribution and a discriminative model D that estimates the probability that a sample came from the distribution of the training data rather than the output distribution of the generator G. The training process for G is to minimize the probability that D labels correctly, which leads to a possible solution where G replicates the training data and D predicts $\frac{1}{2}$ everywhere. The training of GANs is complex and suffers from many issues, ranging from instability to mode collapse [10]. Mode collapse is a phenomenon that occurs when a random input provided to the generator creates highly correlated images. It is common for the model to oscillate between modes while training [10]. Another typical issue with GANs is that the discriminator becomes far superior, leading to vanishing gradients and halting the training progress of the generator [10]. Additionally, many researchers are compute constrained, desire large images, and lack the large amounts of VRAM required for large batch sizing, further contributing to mode collapse. One way to combat this is to use gradient accumulation between multiple batches, but this dramatically slows down training times.

More concretely, G and D can be thought of two opponents in a minimax game following the equations from Torralba et al. [11]:

$$\underset{G}{\operatorname{argmin}} \max_{D} \mathbb{E}_{\mathbf{x}\sim p_{data}}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}}[\log(1 - D(G(\mathbf{z})))] \tag{1}$$

For clarity, it is possible to decompose the minimax game into distinct objectives for D and G.

$$D^* = \underset{D}{\operatorname{argmax}}\, \mathbb{E}_{\mathbf{x}\sim p_{data}}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}}[\log(1 - D(G(\mathbf{z})))] \tag{2}$$

$$G^* = \underset{G}{\operatorname{argmin}}\, \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}}[\log(1 - D^*(G(\mathbf{z})))] \tag{3}$$

From these equations we can directly that the loss function is zero-sum and can cause issues when one side, typically the discriminator, becomes too powerful, no matter what the generator does it will always get a gradient of zero halting training. In this scenario, the generator uses its generative capacity to produce one image very accurately, ignoring other mode collapse.

2.1.1 BigGAN

BigGAN, designed by DeepMind, can synthesize images from 128x128 to 512x512 was the previous state-of-the-art in generating conditional class images improving the state-of-the-art Inception Score from 52.52 to 16.65 on ImageNet. BigGAN improves upon state-of-the-art by using a substantially larger generator and discriminator networks and a better regularization scheme [1]. BigGAN, however, is noted to be unstable, without a significant performance cost.

2.1.2 StyleGAN + StyleGAN-XL

StyleGAN-XL is a state-of-the-art generative model that can produce images up to 1024x1024 on large, diverse datasets [9]. StyleGAN-XL is an iteration of the StyleGAN architecture, first introduced in 2018 by [6]. StyleGAN intern is a standard GAN design described in subsection 2.1. However, an alternative generator network is motivated by a style-transfer network. StyleGAN also incorporates some training tricks that help fix mode collapse and failure of convergence. These tricks include using a Baseline Progressive GAN [4], bilinear upsampling, adaptive instance normalization, style mixing, and insertion of noisy inputs. An ablation study is conducted to assess the importance of each of these methods in [6]. StyleGAN-XL also automatically train super-resolution layers, reducing the generator saving computation complicity.

2.2. Image super-resolution

Image super-resolution (SR) is a subset of image restoration, aiming to output high-resolution images from low-resolution input images. This allows us to take a low-fidelity or noisy image and generate a higher quality image that performs better than non-model-based upsampling such as bilinear upsampling. In this paper, we will be comparing four different pre-trained image superresolution methods and determine which model works the best on NFTs; SWIN-IR, BSRGAN, and Real-ESRGAN [7, 12, 13].

2.3. CLIP – Connecting Images and Text

Contrastive Language–Image Pretraining (CLIP) [8] is a neural network proposed by OpenAI to learn representations for images and text jointly. CLIP uses both an image encoder (e.g., ResNet/Vision Transformer) and a text encoder (e.g., GPT-2) and uses contrastive pretraining on paired image-text data to embed images and text in the same space. In this project, we do not explore CLIP in detail and primarily use a pre-trained CLIP model as a black box to assess the similarity of a generated image compared to an input image and text.

3. Approach

The aim of our proposed pipeline is to design an end-to-end pipeline for generating NFT collections with limited compute resources, i.e. a single GPU and limited time constraints i.e. the whole pipeline is trained in less than a few days maximum. Our pipeline includes five distinct steps.

- Build a web scraper for NFT collections from OpenSea.
- Downsample images
- Train GAN on downsampled images
- Generate images by exploring the latent space, using CLIP to guide the generation process.
- Upsample the images using an image super-resolution network

3.1. NFT data collection

Initially, we web scraped OpenSea using Selenium to compile a database of NFT images. This method is not robust and frequently crashes, leading to missing data sections. We obtained an OpenSea API key online through this OpenSea's API application form. The OpenSea API allowed us to robustly request full-resolution images of any collection of interest at high throughput. Initially, we downloaded the following NFT collections.

- Kreepy Club (9,999 images, evenly split between 3 creators with unique artistic styles, $6.5m USD [3])
- Azuki (10,000 images, same artistic style for all images, $6.5m USD)
- Bored Ape Yacht Club (10,000 images, same artistic style, $645m USD)
- Haki NFT (5,000 images in a similar artistic style as Azuki, $1450m USD)
- CryptoPunk (10,000 images in a 8-bit art style, $2348m USD)

3.2. Determination of input image size

Our goal is to train an end-to-end platform for NFT generation. Therefore we would like to produce the highest resolution images possible given our hardware, one NVIDIA RTX 2080TI, with 11 GB of GDDR5. This represents a significant challenge when many GAN architectures are trained using at least 8 GPUs clusters.

For each architecture, we investigated what the highest supported resolution would be given a batch sizing of at least 8. It was determined for each architecture that 128x128 was the largest we could support. This realization required us to add the additional need for SwinIR to upsample our images [7].

Furthermore, because of limited compute resources and our aim of training an end-to-end NFT generation pipeline using a single GPU within a matter of days, we used down-sampled 128x128 and 256x256 versions of the original NFT images as the training data. These sizes were selected based on two metrics. First, using BigGAN and StyleGAN's approximate training time, 128x128 images would take approximately 16-32 days to train using BigGAN [4] and training StyleGAN2 on 256x256 images would take approximately 32 days to train [5]. We also qualitatively assessed the capacity for different down-sampled images to be reconstructed using a pre-trained image super-resolution network and decided that 128x128 images were the minimum image size that retains texture and fine-level details in the NFT when upsampled (See appendix Figure 9). Since most NFTs have smooth textures, down-sampling images to 256x256 affect quality minimally. A GAN is trained on the low-resolution version of the original images. Then a pre-trained image super-resolution network, SWIN-IR, is used as a black-box upsampler on the final generated images. It is possible to train a custom image super-resolution module. However, due to time, compute resources, and the diversity of the different NFT collections used in this project, a pre-trained super-resolution module is preferred.

Because GANs are difficult to train, we aim to use down-sampled 128x128 versions of the original NFT images as the training data. Then, we pass these low-resolution generated images through a pre-trained SR model to reconstruct a higher resolution 512x512 output image using an SR-GAN network and treat it as a black-box [7]. Due to limited computing resources, we train each GAN separately on each NFT collection to reduce the time needed for convergence and reduce the chance of mode collapse during the training process.

3.3. Training the GAN

Our approach considered many different GAN architectures for generating NFTs with different specifications. In this project, we explored four different architectures such as BigGAN[6], StyleGAN-XL[7], VQGAN[8], StyleGAN2-ADA[9] as architectures for generating NFT [1, 2, 5, 9]. Their im-

[3] Volume traded in $USD using 1ETH = $2600USD

[4] The orignal authors of BigGAN [1] estimate 24-48 hours to train using 128 TPU cores on 128x128 images, which equates to 24 days of training time on the 8 TPU cores available on Colab.

[5] According to the original authors estimates available at https://github.com/NVlabs/stylegan2

[6] https://github.com/ajbrock/BigGAN-PyTorch

[7] https://github.com/autonomousvision/stylegan_xl

[8] https://github.com/CompVis/taming-transformers

[9] https://github.com/NVlabs/stylegan2-ada-pytorch

plementations are readily available through each author's GitHub repository.

Our original approach was to use BigGAN and StyleGAN-XL exclusively since they are state-of-the-art models. However, during the project, we realized that these model architectures were not designed for NFT collections. NFT generation is highly irregular compared to natural image generation using ImageNet, CIFAR, FFHQ, CelebHQ, etc. This is because NFTs have a very regular global spatial structure, very smooth colors, limited variation in textures, and very low noise levels compared to natural images.

This section considers different ways of training GANs and describes some shortcomings of using GANs for generating NFTs. These architectures were implemented sequentially throughout the project as we searched through different methods to solve the failure modes of GAN training on our data.

After the ground truth and NFT data sets were compiled, we center-cropped non-square images and downsampled the source resolution to 128x128 using OpenCV and Lanczos interpolation.

3.4. BigGAN

Our first attempt was to train BigGAN on Cryptopunks, one of the most simple NFT collections (see Figure 1a). We used BigGAN-Tensorflow as an starting point [1].[10] After a significant amount of hyperparamater tuning, such as changing the generator learning rate, loss function and Adam beta parameters, we found that BigGAN was too unstable for our small batch sizing of 64 128x128 images and would constantly diverge as seen in Figure 1c. Based on these results we decided to retire BigGAN and move to a more contemporary architecture that is more robust.

3.5. Stylegan-XL

Since StyleGAN-XL has the ability to train a class conditional dataset and is shown to work on large diverse image datasets. We attempted to train a model that learns how to generate 12 NFT collections and 6 wiki-art collections simultaneously. Each collection was treated as its own class. Since we could only manage a batch size of 12 using our compute resources not even one example per class could be optimized over per batch, directly leading to complete mode collapse shown in Figure 2. After training for 120 hours, progress was no longer being made and we decided to adjust our approach to handle each NFT collection individually. Another approach would be to use gradient accumulation to artificially increase our batch size, however this would slow down training time by a factor proportional to the amount of gradient accumulation. We also trained StyleGAN-XL with default hyperparameters on the Azuki collection, which performed better than training on a large class-conditioned dataset, however still faced the issue of mode collapse.

The problems we faced when using these to models on small subsets of data are

- Training on multiple collections of class-conditioned datasets requires too much computational power given our limited resources
- There is little semantic similarity between NFT collections, thus leading to mode collapse within each collection.
- NFT collections are highly structured. Images have very similar spatial structure and limited texture variation within a collection, leading to the discriminator easily overfitting.
- Training on individual datasets (less than 10k images) always leads to mode collapse as the discriminator and generator for BigGAN and StyleGAN-XL are too expressive, even under different learning rate specifications.

3.5.1 StyleGAN2 with Data Augmentation

To combat the issue of mode collapse on small datasets, we resort to GANs with lower expressive power than the architectures considered above. Furthermore, we focus on GAN architectures designed to generate images from small datasets. StyleGAN2 with Adaptive Discriminant Augmentation (SG-ADA) is a GAN architecture that is designed to generate images from a small set of training images.

For our architecture, we used a scaled-down version of the original StyleGAN2 architecture that better captures the capacity of our downsampled images. This is done by halving the size of the feature maps within the network as suggested in [5]. This has the benefit of faster training and reduces the probability of overfitting. We also select a subset of the data augmentations used in SG-ADA that are most relevant for NFT training. In particular, we only use color transformations, frequency filtering, and additive noise augmentations. Other augmentations such as horizontal flip and cutout are not suited for NFT training as images within a single NFT collection that we considered always face the same way.

Two other discriminator regularization mechanisms are paramount to our approach to avoid mode collapse during the training process. The first is to adjust the $R1$ regularization parameter during training to $R_1 = 200$ (or γ) to slow down the discriminator gradient updates when observing the true data to delay the discriminator from overfitting. The default heuristic for setting the R_1 parameter is given by $\gamma_{heuristic} = 0.0002w \cdot h/B$ with a suggested range of $[\gamma_{heuristic}/5, \gamma_{heuristic} \cdot 5]$, where w is the width and h is

[10] https://github.com/taki0112/BigGAN-Tensorflow

the height and B is the mini-batch size. This leads to initial value of $\gamma = 0.8192$ which is far too little regularization for our dataset and leads to the discriminator overfitting and mode collapse during the training process.

The second approach is to delay the discriminator from updating for the first 10,000 steps as proposed in the implementation of [2]. Furthermore, we use an increased learning rate and smaller batch size during the first half of the training process for faster convergence before moving to a default learning rate and batch size.

We believe that our solutions to the GAN training process are incredibly dependent on the type of data we are training on. NFT data behaves very differently from natural landscapes and faces that most GANs are fine-tuned to recreate. Specifically, due to the low amounts of spatial and textural variation within a single NFT collection, the discriminator learns much quicker than the generator, leading to mode collapse. The bag of tricks we implement here takes from many ideas we have seen across different GAN implementations.

With a mini-batch size of 16 using 256x256, training on one GPU using mixed-precision training, we achieve a training speed of 23.30 seconds per 1000 images (k/img) with a total GPU memory load of $< 8GB$ as measured using the NVIDIA System Management Interface. The approximate training time for 5000 k/img iterations is 32 hours, which is well within a feasible computational budget for training GANs.

We used a smaller capacity generator and discriminator network compared to the original StyleGAN2, where the feature maps in the synthesis network are size 128 instead of 512 as in the original StyleGAN.

3.6. Using CLIP to guide generation

CLIP is used as a black box to assess the least distance between an input image or text chosen by the user and the generated image from our trained generator. To generate an image that has the most semantic similarity to the inputs, we used the L_2 loss function between the CLIP embeddings of our generated image and the input image. We then use this score as a loss to optimize the latent vector that is used to generate our image. To do this, we first embed the guide image using CLIP. Then we embed the generated image using CLIP and calculate the L_2 loss between the embeddings of the generated image and the guide image. We then use the Adam optimizer with a learning rate of 10^{-3} and optimize over the latent-space vector to generate an image that is closest to the guide image. Different types of loss functions between embeddings can be used to score the similarity between the guide image and the generated image, such as the L_1 loss or cosine similarity. The CLIP model that we used was a pretrained CLIP model with ResNet-50 from OpenAI.

4. Results

4.1. Comparison of GAN architectures

Figure 4 show the generator and discriminator losses at different steps of the training process for the StyleGAN-XL and StyleGAN-ADA models that we trained in the project.

There are two clear patterns to these plots. First, in Figure 4 we can see that the StyleGAN-XL models all fail to converge. The generator loss is very large, and the discriminator loss is very small, a typical sign of mode collapse as the discriminator loss, and generator loss diverge. The discriminator loss drops to values close to zero at the $\lesssim 300$ steps. The StyleGAN-ADA model with R_1 regularizations of 100 and 400 both display good characteristics

Figure 5 show the Fréchet Inception Distance (FID) at different steps of the training process and also at different wall times during training. The FID score is the squared Wasserstein metric between two multidimensional Gaussian distributions. This is a way of comparing the statistics of generated samples to real samples and is a commonly measured metric to score the quality of a trained GAN [3], a lower score indicates better quality generated images.

The FID scores for StyleGAN-XL trained on the combined NFT collections, Azuki collection, and Haki collection failed to decrease over the training process. StyleGAN-ADA trained on the Haki collection for R_1 values of 100, and 400 show a decreasing FID score throughout the training process. Moreover, since we used a smaller generator and discriminator network, the training process is faster. The lowest FID score is achieved by our StyleGAN-ADA network with $R_1 = 100$, with an FID score of 13.2 in 32 hours of training time on a single GPU. When training StyleGAN-XL on the Azuki collection for 72 hours, the lowest FID score achieved was 67.66.

4.2. Image Super Resolution

To upsample the 128x128 and 256x256 NFTs created by the GANs to 512x512 and 1024x1024, respectively, using one of four pre-trained image super-resolution models SwinIR, SwinIR-Large, BSRGAN, and Real-ESRGAN [7, 12, 13]. We found qualitatively that SwinIR-large generally performed the best and preserved detail without applying too much smoothing to the image. BSRGAN tended to shift the style of the image to look as though it is a watercolor painting and smoothed the image substantially more.

4.3. Image generation

Figure 6 show the results of our image generation procedure using a picture of Donald Trump as a target image. We can see that the generated image mimics some of the qualities in the original image. Namely, the generated image has white hair, yellow portion in the background, open mouth, and similar clothing.

Figure 7 shows the results of our image generation using the text input 'blue shirt pink hair' as a target text. The figure shows that the generated image after the optimization procedure matches the target description even though the initial images do not resemble the target text.

5. Conclusion

In conclusion, we have created an end-to-end system for generating NFT's from images and text at high resolution. In the project, we train multiple GANs on NFT collections and address the failure modes of training GAN's on these types of images. Our final model achieves an FID score of 13.25 and takes $\lesssim$32 hours to train on a single GPU. We do this with computationally feasible budget and our final model's training time takes around 32 hours on one GPU. We envision our model having applications to avatar generation based on input images or text.

5.1. Potential ethical concerns of the project

We uncovered some ethical concerns during this project, which we will discuss in this section. There are two main ethical concerns that we discovered. First, the property rights of NFT collections are unclear at this stage, and training a neural network to reproduce an individual's artwork may pose ethical issues. However, due to the anonymity of NFT ownership through the blockchain, there is a limited legal risk of reproducing artwork. Second, we see these systems as potentially harmful, especially in generating digital artwork that most closely resembles an input image. There are potential use-cases to inflict harm to individuals by generating derogatory artwork of a person, accentuating unpleasant characteristics of a person, or artwork that caricatures offensively.

5.2. Future work

In the future we would like to improve our pipeline by creating a web interface where people can log in and automatically generate the images they need for a small subscription fee. This could not only be useful for NFT generation but for stock photos and marketing resources. We would also like to get access to more computational resources to train a class-conditional network that does not exhibit mode collapse.

6. Contributions

Overall, we equally contributed to this project and note that the major contributions made by each person was.

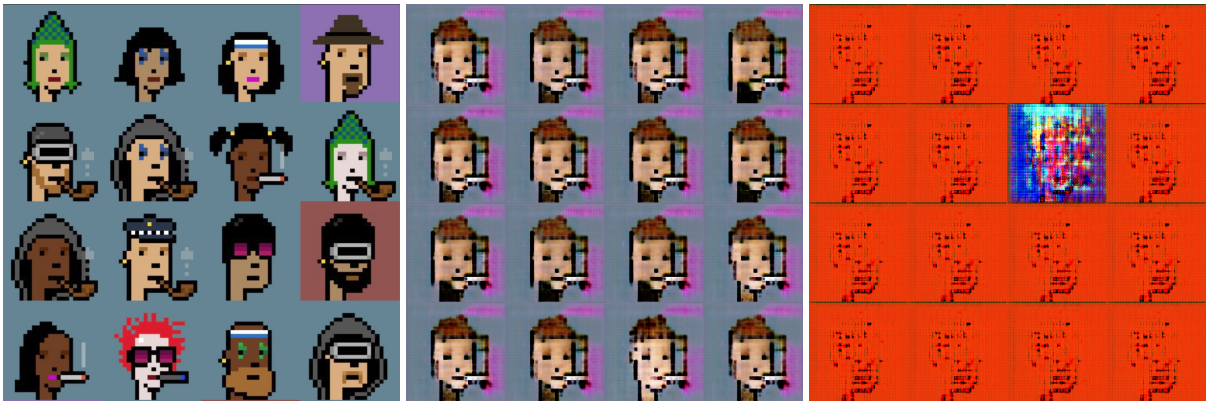
Contributions made by Alex Kashi were:

- Web scraping OpenSea
- Compiling a Dataset for all NFTs
- Training BigGAN
- Training Stylegan-XL on (All NFTs, Apes, Haki, and Azuki)
- Obtaining training metrics
- Writing half the report

Contributions made by Geoffrey Liu were:

- Investigating potential of mode collapse
- Training StyleGAN-ADA on Haki collection and generating outputs from the model.
- Using CLIP to guide the generation of images
- Exploring the latent space of the GAN.
- Implementing the image super-resolution model
- Writing half the report and plotting loss and FID scores.

All work that isn't listed in the contributions was equally divided.

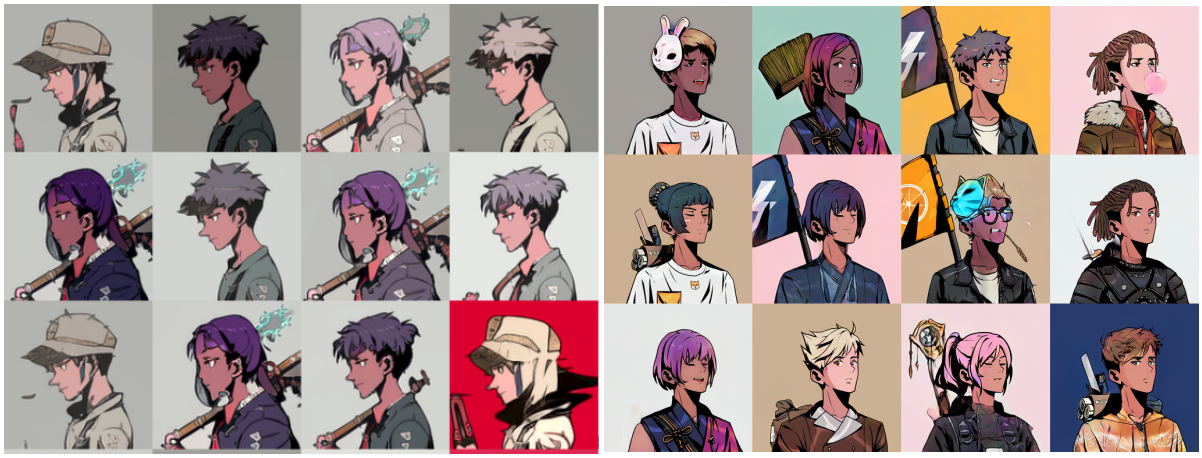


(a) CryptoPunks Original (b) CryptoPunks BigGAN Training (c) CryptoPunks BigGAN Diverge

Figure 1. BigGAN training on Cryptopunks dataset, results in collapse, due to inferior unstable architecture and small batch size

(a) NFT Original Collections (b) Collapsed StyleGAN-XL after 0.5 kimg

Figure 2. StyleGAN-XL class conditional inputs (a) and outputs (b) for both figures columns 1-5 are part of one collection while columns 6-9 are part of another collection, and each row is a separate collection.



(a) Example StyleGAN-XL output trained on Azuki collection for 72h, with a FID score of 67.66 (b) Example StyleGAN-ADA output trained on Haki collection for 32h, with a FID score of 13.23.

Figure 3. Comparison of generated images from StyleGAN-XL and StyleGAN-ADA. We can see that StyleGAN-ADA generates higher quality and much more diverse images with less training time.

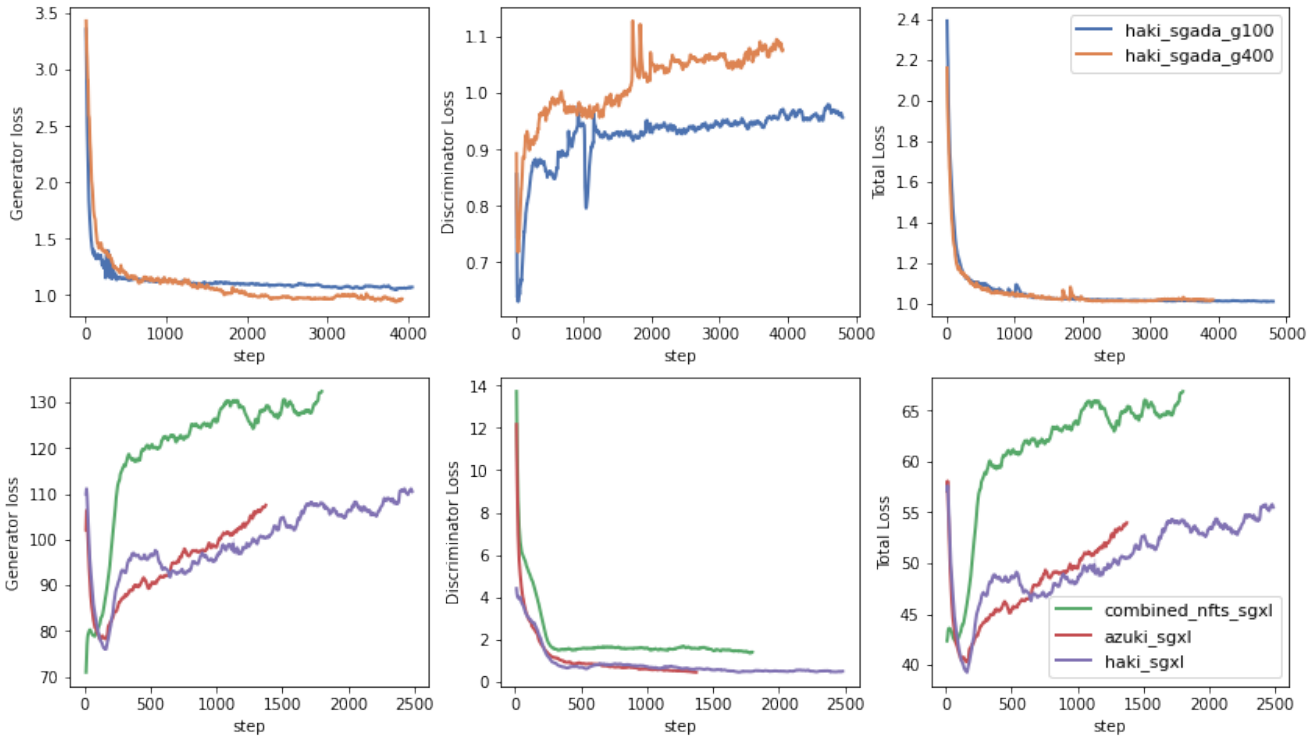


Figure 4. Discriminator, generator and combined loss during the training process for different GAN architectures. We can see that the StyleGAN-XL models exhibit failure modes in training, the generator loss increases quickly and the discriminator loss decreases to 0 very fast. StyleGAN-ADA for both $R1$ specifications of 100 and 400 exhibit better loss progression throughout training as the loss of the generator and discriminator are balanced.

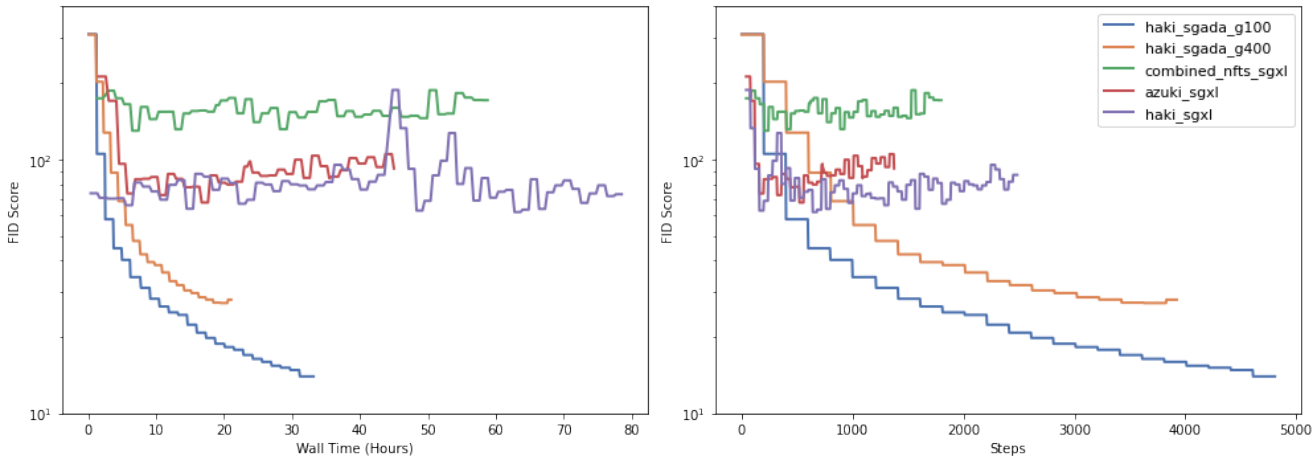


Figure 5. Fréchet Inception Distance scores for different GANs versus the number of steps. Each step is equivalent to training on 1000 images. We show the wall time of the different models we trained in hours on the left plot. We see that SG-ADA outperforms SG-XL in both FID and FID scores per step. We can see that SG-ADA converges faster with $R_1 = 100$, indicating that $R_1 = 400$ may over-regularize the discriminator. SG-XL on the combined NFT dataset, Azuki, and Haki all fail to converge and experience mode collapse during the training process, which can be seen by the failure of the FID scores to decline during the training process.

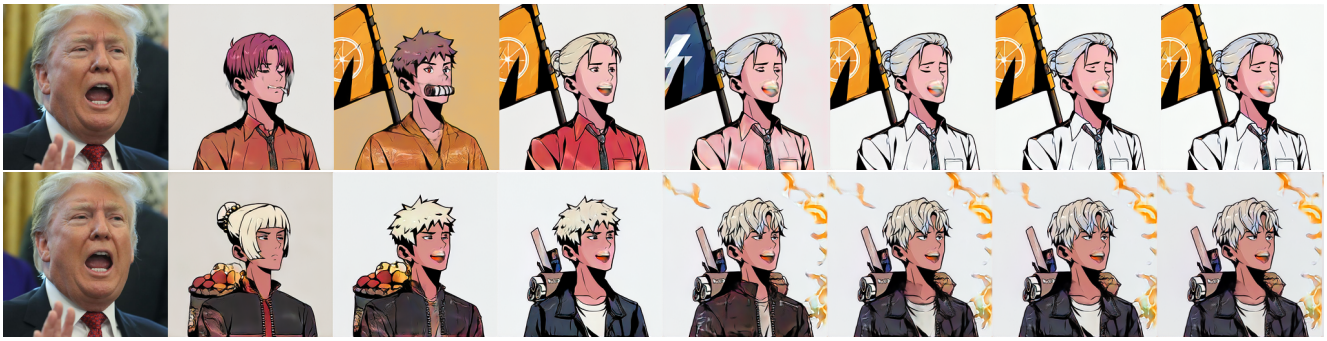


Figure 6. Image generation using CLIP and a picture of Donald Trump as the target image. The first image from the left is the target image, and the second image is the generated image from the GAN at initialization. The rightmost images show the output image at the end of the optimization procedure. The images in between are evenly spaced snapshots over the optimization steps.

Figure 7. Image generation using CLIP and the first row uses the text target 'male blue shirt pink hair' and the second row uses the text target 'female blue shirt pink hair.' The leftmost image is the generated image from the GAN at initialization. The rightmost image shows the output image at the end of the optimization procedure using the 1000 steps of Adam. The images in between are evenly spaced snapshots over the optimization steps.

(a) 256x256 GAN output (b) 4x BSRGAN [13] (c) 4x Real-ESRGAN [12] (d) 4x SwinIR-Large [7]

Figure 8. A comparison of different image super-resolution techniques for up sampling 256x256 images by 4x

References

[1] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis, 2018. 2, 3, 4

[2] Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12873–12883, June 2021. 3, 5

[3] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. 5

[4] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation, 2017. 2

[5] Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training generative adversarial networks with limited data, 2020. 3, 4

[6] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks, 2018. 2

[7] Jingyun Liang, Jiezhang Cao, Guolei Sun, Kai Zhang, Luc Van Gool, and Radu Timofte. Swinir: Image restoration using swin transformer, 2021. 2, 3, 5, 9

[8] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021. 2

[9] Axel Sauer, Katja Schwarz, and Andreas Geiger. Stylegan-xl: Scaling stylegan to large diverse datasets. *arXiv preprint arXiv:2202.00273*, 2022. 2, 3

[10] Divya Saxena and Jiannong Cao. Generative adversarial networks (gans) challenges, solutions, and future directions. *ACM Computing Surveys (CSUR)*, 54(3):1–42, 2021. 2

[11] Antonio Torralba, Phillip Isola, and William F. Freeman. *The Tiny Book of Computer Vision*. MIT, 2020. 2

[12] Xintao Wang, Liangbin Xie, Chao Dong, and Ying Shan. Real-esrgan: Training real-world blind super-resolution with pure synthetic data. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1905–1914, 2021. 2, 5, 9

[13] Kai Zhang, Jingyun Liang, Luc Van Gool, and Radu Timofte. Designing a practical degradation model for deep blind image super-resolution. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4791–4800, 2021. 2, 5, 9

A. Appendix

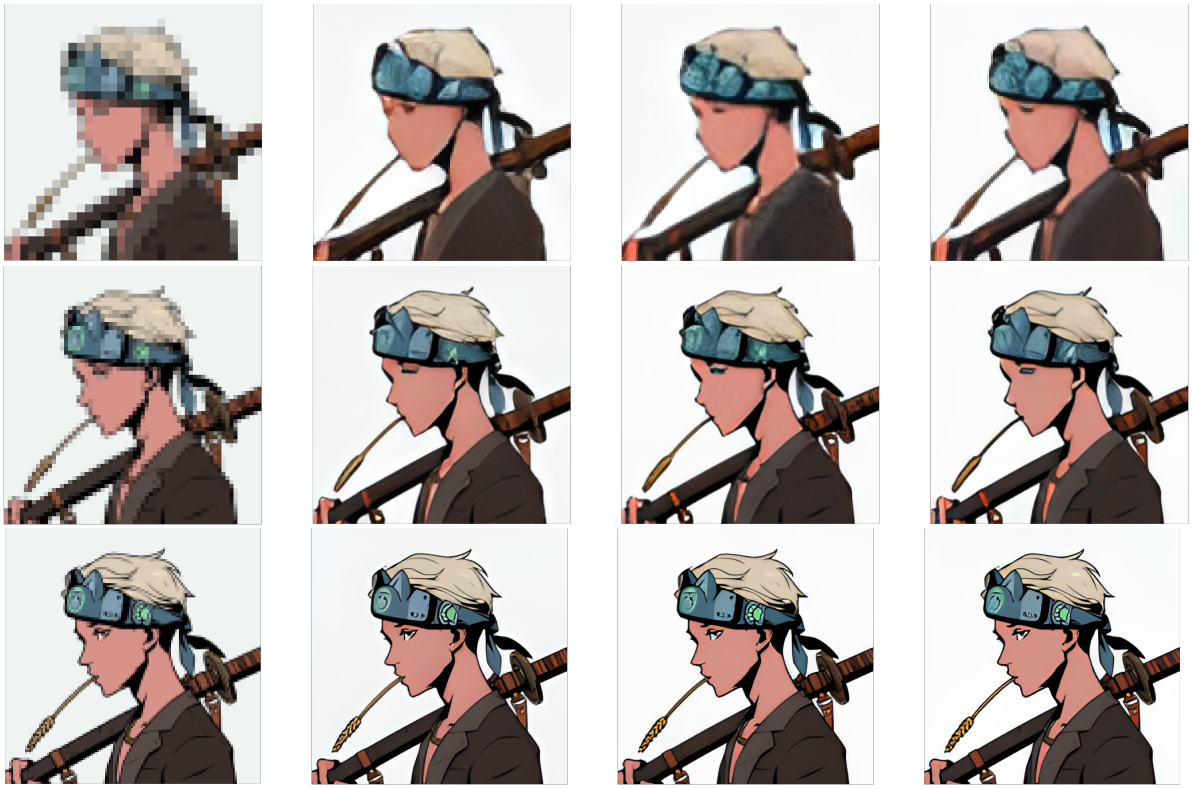


Figure 9. Supplementary figure: Comparison of image superesolution on different resolution input images. The top row uses a 32x32 input, the middle row uses a 64x64 input and the bottom row uses a 128x128 input. The plot on the left is the input image, the second plot from the left shows upscaled outputs using ESRGAN, the third plot from the left is shows upscaled outputs using SWIN-IR and the rightmost plot shows upscaled outputs using SWIN-IR XL. Each input image was upscaled to 4x the input size.