Creating Pixelated Characters Using GAN

Yiming Li (billyli)
billyli@umich.edu

Tianyuan Xu (tyxu)
tyxu@umich.edu

Jiahao Fu (jiahaof)
jiahaof@umich.edu

Muyi Zhang (muyiz)
muyiz@umich.edu

Abstract

Image generation is a process of learning features from the given training data and creating new samples with the similar features. In this article, we utilize Generative adversarial network (GAN) to generate new non fungible token pixelated characters. We then modified the existing network to a conditional GAN so that the features of the outputs can be controllable. After training, the proposed network can generate character images that have the features of an input label type.

1 Introduction

An NFT is a non fungible token, unique tokens that can be traded and exchanged on the Ethereum blockchain. NFTs come into many forms: fashion design, music, video game items or even a piece of code can be made into a NFT and registered on the blockchain. The most popular one of all types of NFTs is art which also has the highest market value. The goal of this paper is to generate new digital art based on the current popular NFTs. Among NFT digital arts, pixelated character is a prevalent trend. For model evaluating perspective, pixelated character gives a good visual representation for human eyes to easily decide whether the generated output is valid.

We choose CryptoPunks, the very first NFT series to catch popular interest and one of the most actively traded today, to be our training samples. It is a collection of 10,000 unique collectible characters, each is a 24x24 pixel, 8-bit-style unique character. The market capitalization of all 10,000 CryptoPunks is estimated at around $5.1B by November 2021.

For the scope of this project, we are interested in the type of each character. Characters can be categorized into 5 types: female, male, zombie, ape, and alien. The price of each type varies by much. From Figure 1, zombie type has the highest median price sold. However, CryptoPunks dataset is unbalanced in type. Out of 10,000 characters, 6039 of them are male, 3840 of them are female, and the rest 121 images are zombie, ape and alien. The scarce dataset for zombie, ape and alien makes training a network on them infeasible. Therefore we mainly focus on male and female characters.

In the following section, this article presents the brief introduction of generative adversarial network (GAN) and how we utilize it to achieve CryptoPunks character generation. It also presents our modification to the existing GAN to condition the network generation to a given label type. Our results are shown on the later section.

Figure 1: CryptoPunk Types vs. Median Price Sold

2 Model

In recent years, there has been considerable attentions on utilizing Generative Adversarial Networks to synthesize different kinds of images[3], including the pixelated style images which are our main concern in the study. We first reproduce the GAN model in image generation and set the result as the baseline for further improvement.

2.1 Generative Adversarial Networks (GAN)

Generative Adversarial Network (GAN)[2] is one of the most successful image synthesis models in the past few years and it's been proven to outperform traditional CNN in certain areas, and has been broadly used in super resolution[4], image transfer[6], image generation[5] and many other tasks.The training process can be summarized to the competition between two agents, generator and discriminator. The generator G with parameters θ_g maps the noise data to the data space and try to maximize $[log(1 - D(G(z)))]$ in order to deceive the discriminator. While the discriminator, which is also implemented as a neural network with parameters θ_d, examines the samples created by the generator and determines whether they are real or fake and to maximize to probability of giving the right label to these images. They both compete each other and train simultaneously.

The general objective function of the GAN[7] is

$$\min_G \max_D V(D,G) = E_{x\sim p_{data}} log[D(x)] + E_{z\sim p_z(z)}[log(1 - D(G(z)))]$$

where D(x) denotes the probability that x is from the real data set instead of the generated images from the generator. The generator is also trained simultaneously to minimize $[log(1 - D(G(z)))]$.

Our model, specifically, use Deep Convolutional Generative Adversarial Network (DCGAN)[8]. Compared to the original GAN, the DCGANS usually have better performance because of the advantage of CNN dealing images than MLP. The convolution architecture of our model is given by

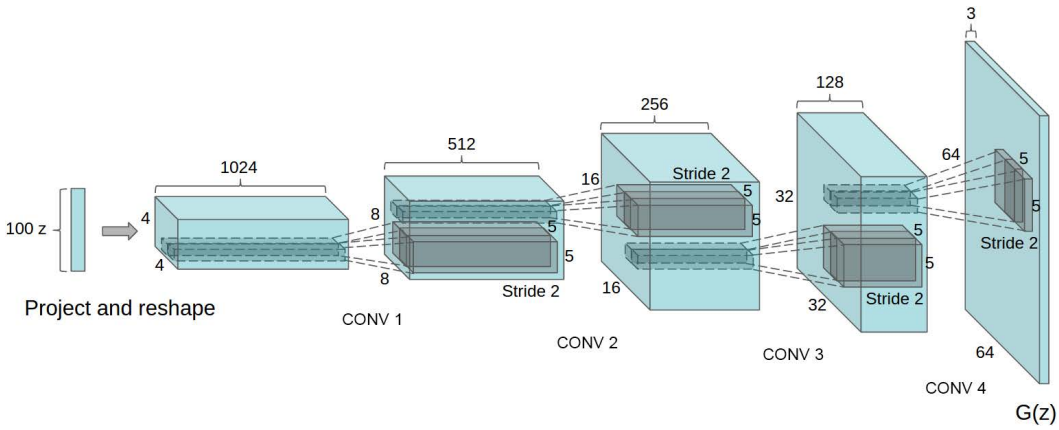

Figure 2: The DCGAN architecture from Radford et al[15].

The general GAN model is proved to be effective in image synthesis and we use this model to generate a group of synthetic pixelated-character images as our baseline for further comparison.

2.2 Conditional DCGAN

The previous DCGAN model is able to generate the general images of pixelated characters from the given data set as a whole. Nonetheless, given the fact that some certain types of the original data has relatively small size with comparatively high value in market, our goal is to find the optimize the generator so that it can generate the desired types of pixelated characters we want.

The objective function of our condtional DCGAN model is given by

$$\min_G \max_D V(D,G) = E_{x\sim p_{data}} log[D(x|y)] + E_{z\sim p_z(z)}[log(1-D(G(z|y)))]$$

where y is the classification information we provide. The structure of our conditional DCGAN model is

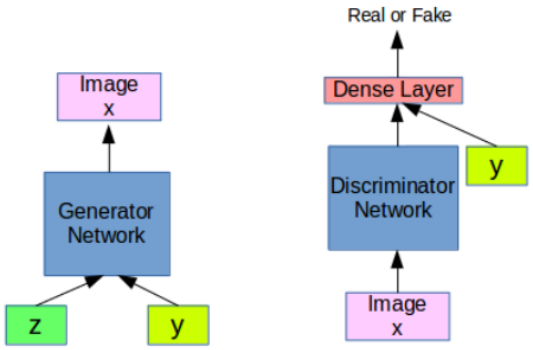

Figure 3: our conditional DCGAN structure.

3 Experiment

3.1 Data Preparation

The data preprocessing is the crucial part of the GAN training. Online dataset platform such as Kaggle provide a large number of open-sourced formatted images that can be used in image generation. The "CryptoPunks" dataset[13] consists of 10,000 PNG images that have different features such as types and price sold.The Cryptopunks are one of the earliest examples of a "Non-Fungible Token" so it's an ideal dataset for our study. The image size is $3 \times 24 \times 24$.

3.2 Hyperparameters

We use a standardized weight initialization model for all of our convolutional, convolutional-transpose, and batch normalization layers. All model weights will be randomly initialized from the normal distribution with mean=0, stdev=0.2 according to Goodfellow(2014)[1]. The learning rate is 0.0002 and we also used Adam Optimizer to optimize all of our models with hyperparameter $\beta_1 = 0.5$. As for the conditional DCGAN, we choose character type as our main classification label y.

3.3 Model Training for DCGAN

In each mini-batch of our dataset, we train the discriminator for one iteration, and then the generator for one iteration. We trained 100 epochs with the batch size = 128.

For the discriminator, we first generated a fixed batch of latent vectors for the generator process. Then train the discriminator using real data and fake images and compute the backward gradients for each respectively. Sum up the loss from real and fake loss and update the optimizer.

The generator, on the other hand, will pass the generated images for discriminator, and then back propagate and update the optimizer.

3.4 Model Training for Conditional DCGAN

A slight difference in conditional DCGAN is the input to both the generator and the discriminator. To condition the output on a given label, the label has to be fed to both the generator and the discriminator. The label is decoded into 5 integer values between 0 and 4 inclusively. Each integer label represents a type of the CryptoPunks character (female, male, zombie, ape and alien).

Before training, each training sample is labeled with a type. During training, we first load the training sample and the label and calculate the score using the discriminator. We then pass the same label and the latent vectors to the generator to output a fake image. The discriminator is used again to compute a score for the fake image. The backpropagation step is the same as the baseline DCGAN.

4 Evaluation Metrics

For the Generative Adversarial Networks, there are many evaluation metrics used to assess the quality of the generated images are IS[10] (Inception Score) and FID[9] (Fréchet Inception Distance, also called the Wasserstein-2 distance), Other commonly used metrics such as Maximum Mean Discrepancy(Kernel MMD) and The 1-Nearest Neighbor classifier. Although some empirical studies[11,12] on how to choose the evaluation metrics for GAN have been done, there's still some debate on this topic. The evaluation metric we apply is the FID, which calculate the distance between the orignal data set and the generated images in the feature space. The FID of the generative model is given by

$$FID = |\mu_1 - \mu_2|^2 + Tr(C_1 + C_2 - 2\sqrt{C_1 C_2})$$

where μ_1, μ_2 are the empirical means and C_1, C_2 are the empirical covariance of the Gaussian random variables from the feature mapping.

5 Results

5.1 Creating New CryptoPunks

Given a list of random noise as the input, the DCGAN model can generate new CryptoPunks which are similar to the training samples, but with their features mixed with each other. From figure.4 and figure.5, the features of the burning cigarette, eyeglasses, and Mohawk haircut are shared and mixed by characters with different skin colors and genders.

Figure 4: Sample pictures from the dataset

Figure 5: New CryptoPunks created by DCGAN

By contrast, the Conditional DCGAN model learns the labels of the characters. Given a desired label, the Conditional DCGAN can generate punk characters that only belongs to that category. However, the Conditional DCGAN needs sufficient sample data as reference to generate punks belonging to a specific label. If some characteristics are rarely labeled among the training samples (e.g., Aliens), the generated picture will be noisy and distorted.

Figure 6: Female (left) and Male (right) CryptoPunks created by Conditional DCGAN

Here we used the sufficiently labeled Male and Female characters as an example. From Figure.6, when the model is given the task of generating male and female characters, it mixed the blond hair to a female face with makeup, and gave the burning cigarette to a male character with green hair.

5.2 Output Quality and Error Analysis

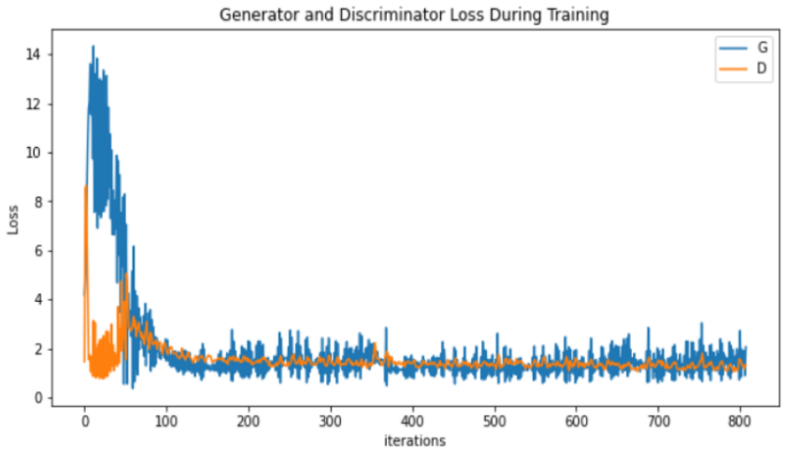

Figure 7: The loss of Generator and Discriminator with respect to iteration

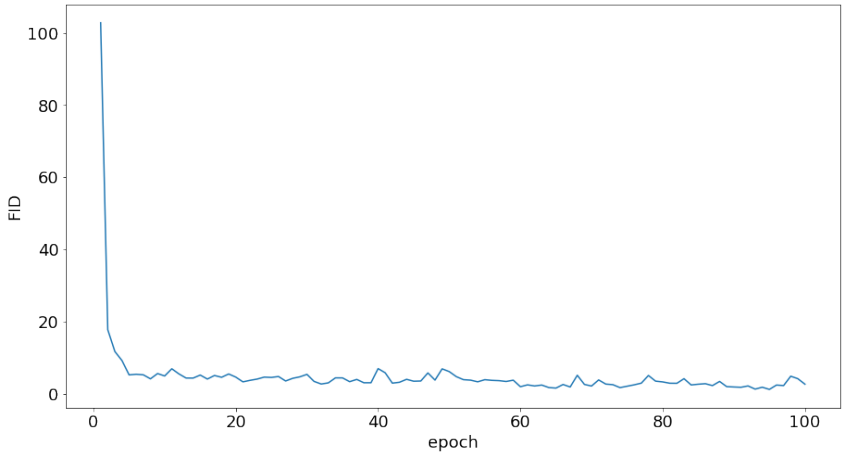

Figure 8: The FID score with respect to epoch numbers

The BCE loss and FID score shows good convergence when the number of epochs become high. But the choice of optimal epoch number also depends on the output quality.

When the model is trained with a low number of epochs, the generator and discriminator noise remain high. The model cannot isolate different features from the characters, thus generates characters with some features overlapping with each other. From figure 8, the character on the left has a brown hat overlapping with a mohawk haircut.

Figure 9: Punk characters generated by an underfitting model

Figure 10: Punk characters generated by an overfitting model

However, when the epoch number is too high, the model tends to generate Punk characters with similar features (e.g., purple baseball caps). When the discriminator is stuck at some local minimum after the training, the best strategy for the generator is to generate the same pattern that escapes the screening over and over. This is called mode collapse in generative models. This is not desired because we want out output to show a diversity of features.

By selecting training samples randomly from the dataset and keeping the epoch number around 40, the model maintains low generator and discriminator loss and FID score. And the output pictures are clear and diverse.

6 Future Work

Firstly, we want to find a way to generate high quality labled CryptoPunks even if the data samples for that characteristic is rare. In the future, we want to try techniques like in [16], where an adaptive discriminator augmentation mechanism to stabilize the training.

It would also be fun to collect our own dataset by internet scraping. We could train our model on a variety of NFT arts, and generate our own arts by simply putting noise and labels into the generator.

7 Group Contributions

All team members worked on evaluating related works and formulating the project ideas. Jiahao Fu collected data and worked on pre-processing. Tianyuan Xu worked on baseline evaluation and result analysis. Yiming Li did research on the algorithm and developed the conditional model. All co-authors equally contributed to this project.

References

[1]Goodfellow, Ian, et al. "Generative adversarial nets." Advances in neural information processing systems 27 (2014).

[2] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Advances in neural information processing systems, pages 2672–2680, 2014.

[3] Jin, Yanghua, et al. "Towards the automatic anime characters creation with generative adversarial networks." arXiv preprint arXiv:1708.05509 (2017).

[4]Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image superresolution using a generative adversarial network. arXiv preprint arXiv:1609.04802, 2016.

[5]Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434, 2015.

[6] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. arXiv preprint arXiv:1611.07004, 2016.

[7]Gui, Jie, et al. "A review on generative adversarial networks: Algorithms, theory, and applications." IEEE Transactions on Knowledge and Data Engineering (2021).

[8]A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," arXiv preprint arXiv:1511.06434, 2015.

[9]M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," in Neural Information Processing Systems, pp. 6626–6637, 2017.

[10]T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," in Neural Information Processing Systems, pp. 2234–2242, 2016.

[11]L. Theis, A. v. d. Oord, and M. Bethge, "A note on the evaluation of generative models," in International Conference on Learning Representations, 2015.

[12]Q. Xu, G. Huang, Y. Yuan, C. Guo, Y. Sun, F. Wu, and K. Weinberger, "An empirical study on evaluation metrics of generative adversarial networks," arXiv preprint arXiv:1806.07755, 2018.

[13]https://www.kaggle.com/datasets/tunguz/cryptopunks

[14]Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.

[15]Radford, Alec, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks." arXiv preprint arXiv:1511.06434 (2015).

[16]Karras, Tero; Aittala, Miika; Hellsten, Janne; Laine, Samuli; Lehtinen, Jaakko; Aila, Timo, "Training Generative Adversarial Networks with Limited Data." .