

NFT Generator

Muhammad Munzer Mardini

Massa Baali

2021-1

Introduction

This project was made to learn Generative Adversarial Networks in a way that was fun and intriguing to me, and that is by trying to generate fraudulent Non-Fungible Tokens that usually can sell to upwards 100,000$ and are practically worthless.

*Keywords: Generative Adversarial Networks (GAN), Non-Fungible Tokens (NFTs)*

1. Dataset

The dataset used was downloaded manually by running a Python script (‘Cyberpunk downloader.ipynb’).

The dataset contains 10,000 colored 24 x 24 px pictures made up of different variants of faces and multiple accessories which made each picture unique.

[Image Number 57
](Cyberpunk/punk0057.png)

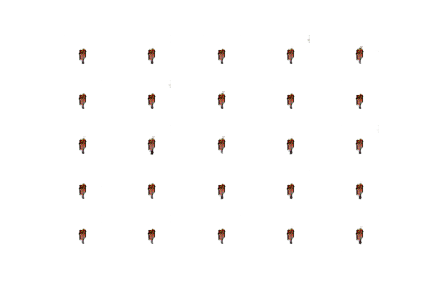
punk0057

24px

24px

1. Pre-Processing

To prepare the images to be inserted into the GAN network it had to be enlarged because applying the filter to a 24 x 24 px image led to the final 1D array being too small to process and resulted in a terrible generated image.

[](Failed%20Attempts/First%20Try/generated_plot_epoch-9.png)

Failed generated image @ epoch

The pictures were then resized into a 64 x 64 px images and kept 4 channels to keep the images colored which was perfect for the Generative network.

1. Architecture

The architecture of the network is split into 3 parts

1. The Discriminative Network:

Which we input the images into to train it, the images are convoluted with a 24x24 filter 4 times and after each convolution it passes a leaky relu and then is flattened and then inputted into the dense layer;

The optimizer used is ADAM with a learning rate of 0.0002 and the weights are updated by using Binary Cross Entropy.

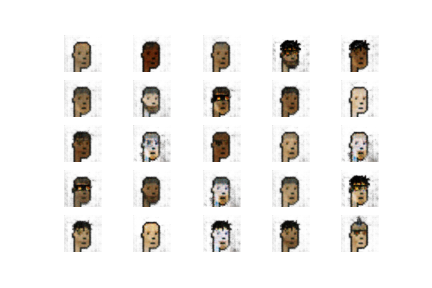
1. The Generator Network:

The first input to the generator network is an array filled with random numbers which is then called the “Noise Input” and then it is inputted into a dense layer that is then de-convoluted with a 32x32 filter 4 times after each time it is treated with a leaky relu until we are left with a 64x64px image which was generated from scratch;

The optimizer used is ADAM with a learning rate of 0.0002 and the weights are updated by using BCE.

1. The last part of the network is combining these two networks together to improve the accuracy of each one to produce a more accurate image
2. Results

This network was trained for 100 epochs which generated decent images with a variety of faces.



Samples @ epoch 20





Samples @ epoch 100

Samples @ epoch 50

The models for the discriminator and generator are saved to improve accuracy or to generate more pictures by using (‘Generator.ipynb’).

100 Generated pictures are in the folder (‘Generated’) which were born using the model that was trained for 100 epochs.

1. How to improve this project

A very important and annoying nitpick is that certain pictures that had a female head or a rare accessory weren’t generated due to their lack of presence in the dataset, this can be improved by tagging each picture with the gender and the accessory in it so that it can be inputted in the network so that the generator network can generate a picture based on inputting what gender or accessory that is wanted.