City University of London

Deep Learning for Image Analysis

Report

Authors:
Elbaraa Elalami

Supervisors:
Dr. Alex Ter-Sarkisov

May 20, 2022

Contents

1 Introduction

The aim of this task is to use artificial intelligence tools to generate art, particularly NFTs. I will be focusing on the BAYC (Bored Apes Yacht Club) dataset, this dataset is a collection of 10,000 unique Bored Ape NFT which are unique digital collectibles living on the Ethereum blockchain. This collection is popular amongst the NFT community, the current floor price (minimum price to purchase one NFT from the collection) of a Bored Ape is 90ETH (215000 USD at the time of writing).

1.1 DCGAN

GANs or generative adversarial networks are a class of machine learning techniques that consists of two networks playing an adversarial game against each other. One of these networks is the generator, the generator can be viewed as some sort of counterfeiter. The second network is the discriminator, it can be viewed as the detective. The networks are simultaneously trained. The training process goes back and forth between the two networks, with the generator creating new samples, and the discriminator trying to predict whether a sample is real or fake.
Given a training set, this technique learns to generate new data with the same statistics as the training set. For example, a GAN trained on photographs can generate new photographs that look at least superficially authentic to human observers, having many realistic characteristics. Though originally proposed as a form of generative model for unsupervised learning, GANs have found application in many other areas such as:

- Text To Image Synthesis
- SRGAN Super resolution GAN: Lower resolution to higher resolution image
- Data Augmentation
- Art: Music generation
- Medical application: Privacy maintenance

DCGANs or deep convolutional GANs were one of the major breakthroughs in the history of GANs [2]. DCGAN at the time of it's publication presented state of the art results, the combination of GANs and convolutional neural networks allowed researchers to improve the quality of images obtained with GANs. In this report I will be exploring one of the simple use cases for DCGANs, which is reproducing samples similar to those in the dataset, particularly pieces of art in the form of NFTs.

1.2 Neural Style Transfer

The key idea behind neural style transfer is that the representations of content and style in the Convolutional Neural Network are separable to some extent. That is, both representations can be manipulated and we can combine the style of one image with the content of another one to produce new meaningful images.

Research showed that when training CNN on object recognition, the network develop a representation of the image such that the input image is transformed into representations that increasingly care about the actual content of the image compared to its detailed pixel values.[4] Higher layers in the network capture the high-level content in terms of objects and their arrangement in the input image while the lower layers simply reproduce the exact pixel values of the original image. Hence, the feature responses in higher layers of the network can be referred

to as the content representation.[4] On the other hand, the style of an image is defined as the degree of correlation between channels of a particular layer. Given these definition, we can set an appropriate loss function in order to produce the desired output.
For this part, the task is to try and replicate the result of paper [3] on the BAYC dataset, that is given a style image (I'm working with famous paintings similar to the paper). What would be the result of transferring the style of such image into the content NFT image.

2 Dataset

2.1 BAYC Dataset

The BAYC dataset contains 10000 images of bored apes, each ape is a (631, 631, 4) png image. Each image has a unicolor background, and contains a unique ape with unique attributes. The dataset also comes with a csv file that maps each image to it's corresponding IPFS hash. IPFS is the InterPlanetary File System protocol, a peer-to-peer network for sharing and storing files. It allow users to store their content in a decentralised fashion, and the content or images in our case can be retrieved from their hashes.

Figure 1: Sample from dataset.

3 Data Prepocessing and Preparation

3.1 DCGAN Dataset Interface

The dataset interface loads the image, then the image is converted to an RGB image with 3 channels. Afterwards, the image is resized to a (64, 64, 3) shape and transformed to a torch tensor. Finally the images are normalised so that each pixel value is between -1 and 1. This is the same procedure followed by Radfort et al.

3.2 NST Datasets

For neural style transfer we have 2 datasets, the original images dataset which is the BAYC dataset and the style images dataset which is a collection of images of old famous pieces of art in the form of paintings. Before training, both images are transformed into a size of $(356, 356)$ and transformed to a torch tensor. Note that 356 is an hyperparameter.

4 DCGAN

4.1 Task and Guidelines

In this part, the aim is to use DCGAN to create fake images of bored apes. The implementation follows the architecture, hyperparameters and guidelines of [2]. This is because GANs have been known to be unstable to train and very sensible to hyperparameters, often resulting in generators that produce nonsensical outputs. The guidelines suggested by the DCGAN paper are as follow:

Model Guidelines

- Use strided convolutions for discriminator and fractional-strided convolutions for generator.
- Use batch normalisation for both discriminator and generator.
- For generator use ReLU, a part from output that uses Tanh.
- For discriminator use Leaky ReLU for all layers.

Weight Initialisation
All weights are initialised from a zero-centered Normal distribution with standard deviation 0.02.

Hyperparameters
Complete list of hyperparameters.

Hyperparameter	Value
learning rate	0.0002
batch size	64
Latent dimension	100
Epochs	25
Image size	64
Adam (β_1, β_2)	(0.5, 0.999)
Leaky ReLU slope	0.2

Model Architecture
The full GAN model is split into 2 models, the generator G and discriminator D. [Fig2]

4.2 Discriminator

The DCGAN discriminator [Fig3] follows a nice structure, the model is structured in blocks. Each block consist of a convolutional layer, batch normalisation layer and a leaky ReLU activation. Note that the batch normalisation layer is skipped in the first block.

The discriminator takes as input a batch of images of shape $(N, 3, 64, 64)$ where N is the batch size. After going through the block of convolutional layers, the image dimension halves each time and the features dimension increases. The final output is then flattened and passed through a fully connected layer with sigmoid activation to obtain a single scalar for each image indicating whether the original image is real or fake.

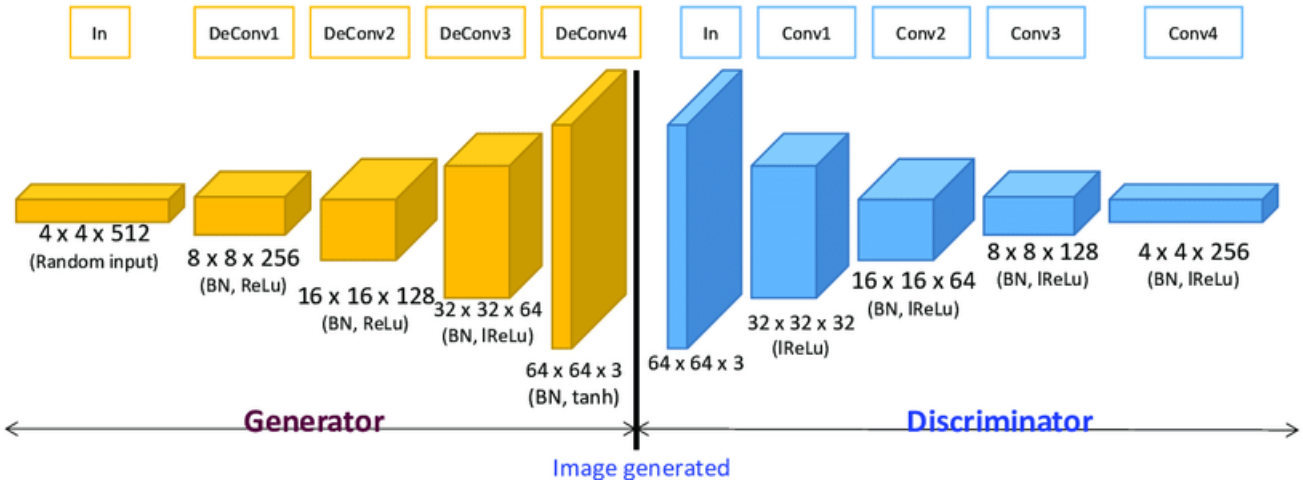


Figure 2: Full model architecture

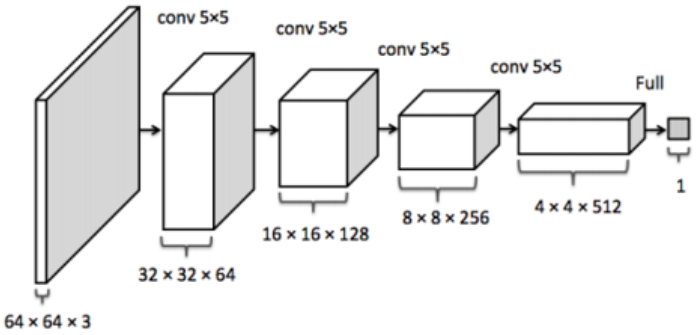


Figure 3: Discriminator Architecture

4.2.1 Discriminator Loss

The discriminator D is trained to differentiate between fake and real images, given a batch of real images and a batch of fake images generated by the generator, the discriminator job is to minimise the binary cross entropy loss of the real images given label 1 and fake images given label 0:

$$L_{discriminator} = \frac{-1}{m} \sum_{i=1}^{m} \left[logD\left(x^{(i)}\right) + log(1 - D\left(G\left(z^{(i)}\right)\right)) \right]$$

Where $x^{(i)}$ are the real samples and $G(z^{(i)})$ the fake ones generated by G.

4.3 Generator

Similar to the discriminator, the DCGAN generator is also structured in blocks, but the difference is that the generator block consist of a transpose convolutional layer instead of a normal convolutional layer, and uses the ReLU activation instead of the leaky ReLU. The transpose convolutional layers are used to upsample the input feature map in order to construct the output image.

The input to the generator is randomly generated noise of size (N, Z_{dim}) where N is the batch size and Z_{dim} is the dimension of the latent space. This input goes through a fully connected layer and then reshaped to the correct size for $(N, 8192, 4, 4)$ for the de-convolutional layers. The final output of the generator is $(N, 3, 64, 64)$ tensor that is passed through a tanh activation to normalise the images to $[-1, 1]$ to have the same range as the original images.

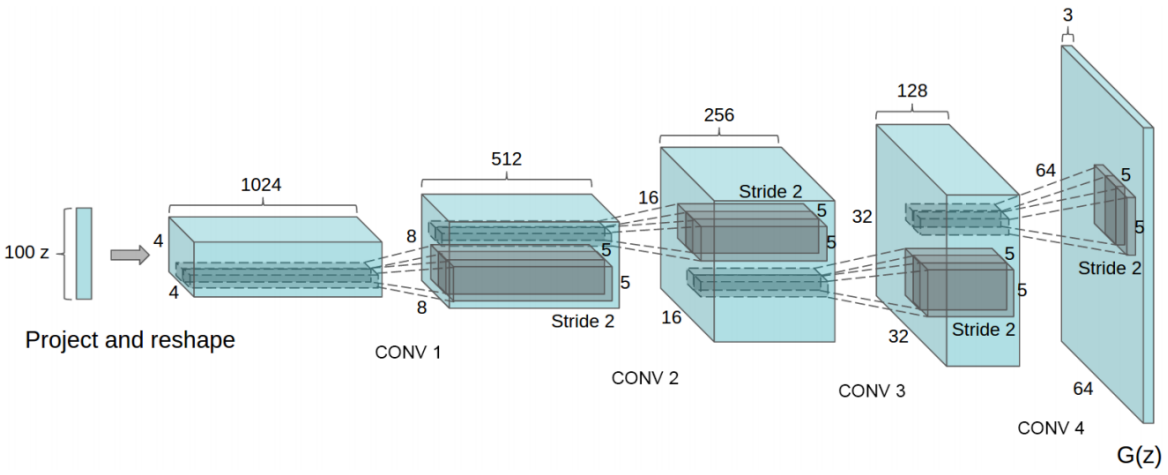


Figure 4: Generator Architecture

4.3.1 Generator Loss

The generator on the hand is trained to fool the discriminator, in order to be successful in his task the generator tries to make the discriminator predict the fake outputs as real. The generator job is to minimise the cross entropy loss:

$$L_{generator} = \frac{1}{m} \sum_{i=1}^{m} log(1 - D(G(z^{(i)})))$$

4.4 Training

The generator and discriminator are trained simultaneously for 25 epochs, in each iteration we perform a step in the opposite direction of the gradient for both models.

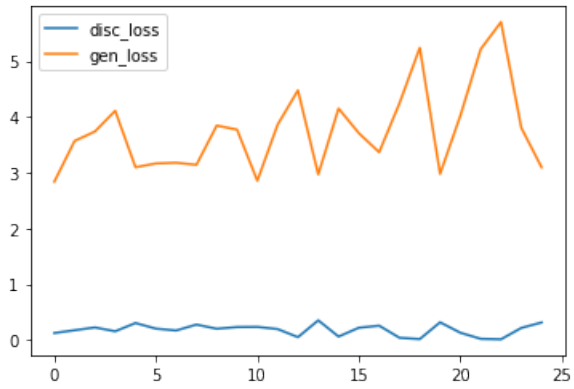


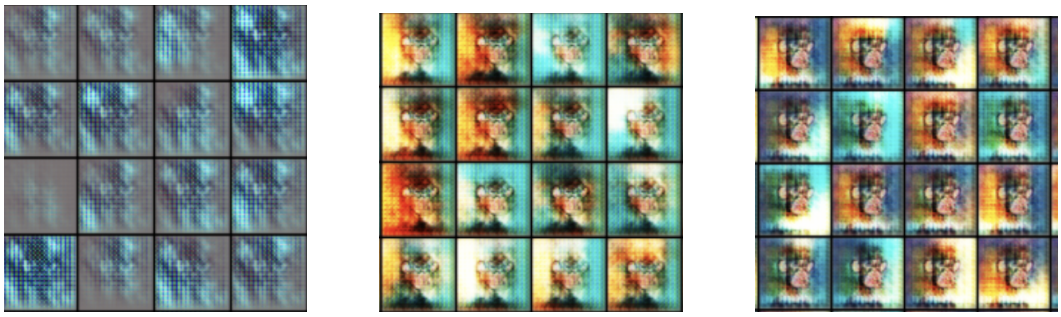
Figure 5: Mean reward during training.

The loss of both the discriminator and the generator goes up and down, from looking at the loss plot we might think that nothing is happening but in the background both D and G are improving, which can be seen in the quality of image produced by the generator.

4.5 Results - GAN Samples

Here I present the images obtained from the generator after running the model for different epochs, we can see that the generator output is improving. In the beginning the images are

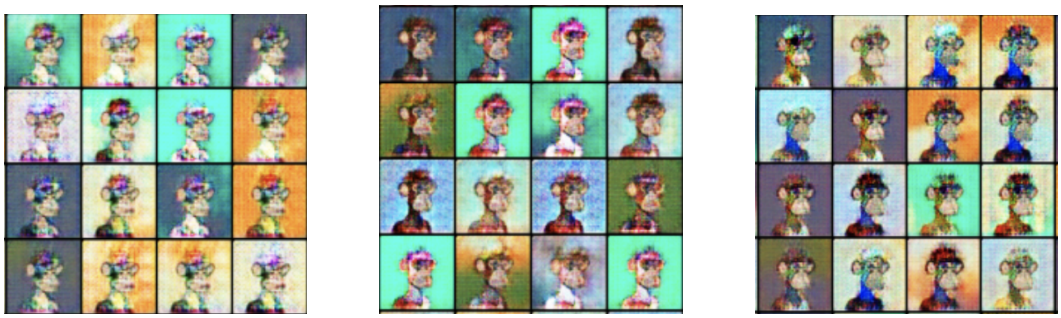
noisy and blurry, we can barely see the form of the ape, with more training the generator gets better and better and the quality of output improves.



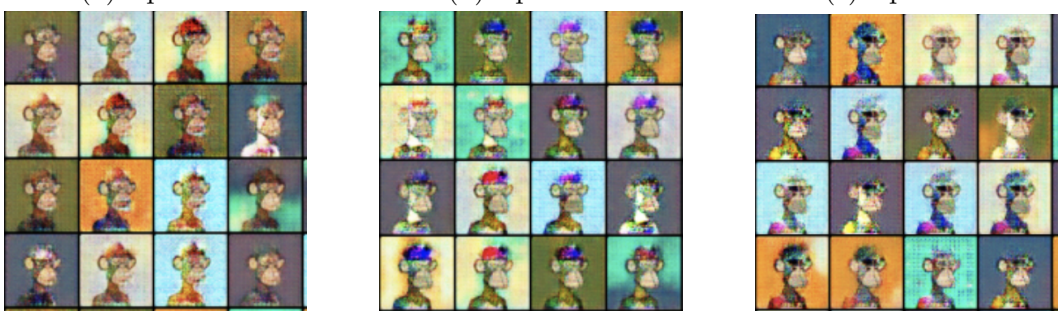
(a) Epoch 1 (b) Epoch 2 (c) Epoch 3

(d) Epoch 4 (e) Epoch 5 (f) Epoch 7

Figure 6: Generated samples at the start of training

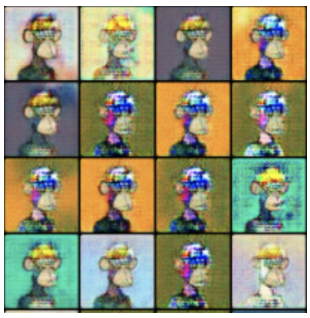


(a) Epoch 9 (b) Epoch 11 (c) Epoch 13



(d) Epoch 17 (e) Epoch 17 (f) Epoch 19

Figure 7: Generated samples at the middle of training



(a) Epoch 21

(b) Epoch 23

(c) Epoch 25

(d) Real samples

Figure 8: Generated samples at the end of training vs Real samples

After a while the model progressively learns the distribution of the data, it learns to draw a unicolor background and an ape in the middle of the picture, although the final output is still not perfect.

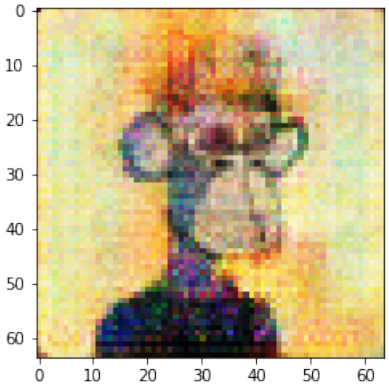


Figure 9: Sample from dataset.

At the end, I generated some samples from the trained model. One of the outputs is presented in [Fig9], the quality of image is not great because we are reshaping the original image in the beginning from $(631, 631)$ to $(64, 64)$ and lose some information. Also, the images are normalised before training and denormalised after obtaining the output which may cause the quality to degrade.

5 Neural Style Transfer

As explained briefly earlier, neural style transfer of NST is a procedure that tries to separate the content and the style of an image. The intuition is by doing so, we can then quantitatively define the difference of content between 2 images and also the difference of style between 2 images. Finally, we can construct a loss function to optimise by having the output image content be closer to the content image and the output image syle be close to the style image.

5.1 Model and Feature selection

The methodology follows the paper[3], first loading the VGG19 model with pre-trained features and freezing all layers. Then, we keep only the first convolutional layers of each block. For VGG19 in python these correspond to layers with index in $[0, 5, 10, 19, 28]$.

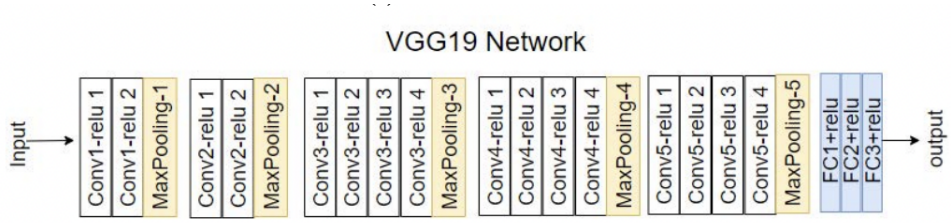


Figure 10: VGG19 Architecture.

5.2 Content Loss

The content loss is defined as:

$$L_{content}(G, C) = \sum_{l \text{ in selected layers}} \left\| a^{[l](C)} - a^{[l](G)} \right\|^2$$

Where G is the generated image and C the original or content image. The content loss simply compares the features of the 2 images for the selected layers.

5.3 Style Loss

The style loss is defined as the norm of the difference between the two Gram matrices of the generated image and the style image. The Gram matrix measure the correlation between activation's across channels, hence if we map the style of the image to it's Gram matrix the style the style loss can be defined as:

$$\begin{aligned} G^l_{i,j} &= \sum_k g^l_{i,k} g^l_{j,k} \\ S^l_{i,j} &= \sum_k s^l_{i,k} s^l_{j,k} \\ L_{style}(G, S) &= \sum_{i,j} \left(G^l_{i,j} - S^l_{i,j} \right)^2 \end{aligned}$$

Where $G^l_{i,j}$ and $S^l_{i,j}$ are the Gram matrices of G the generated image and S the style image. The sum is done over the height and width of each channel.

We know that each channel is responsible for detecting certain feature, the intuition behind this definition of style is that for example if we have a first channel capturing certain texture and a second one capturing the orange colour, the correlation tells us whether the image will or will not have this particular texture with the particular colour (orange).

5.4 Loss Function

The loss function is a weighted sum of the content loss and style loss, it is defined as:

$$L(G) = \alpha L_{content}(G, C) + \beta L_{style}(G, S)$$

Where α, β are hyperparameters, we could use only one hyperparameter since only the ratio $\frac{\alpha}{\beta}$ matters in the optimisation process, but I kept it to stay compatible with paper.

The ratio $\frac{\alpha}{\beta}$ can be explained as measuring how strongly we want to apply style of the style image to the original image.

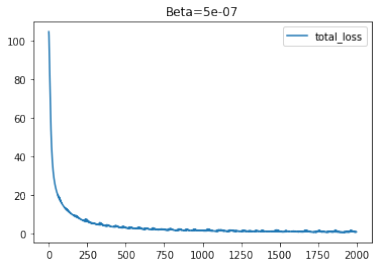
5.5 Training

Unlike the original implementation where the authors started with an image generated by noise, the starting image I used is the original content image. This helps to get good results faster. We perform 2000 update step to the original image, in each step the features of the generated image are modified by the optimiser.
Hyperparameters list:

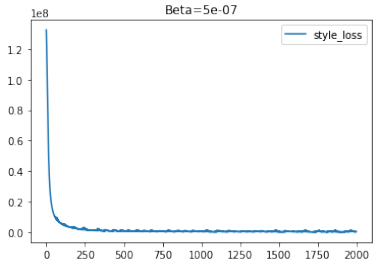
Hyperparameter	Value
learning rate	0.001
α	1
β	0.025
steps	2000
Image size	356

5.5.1 Losses

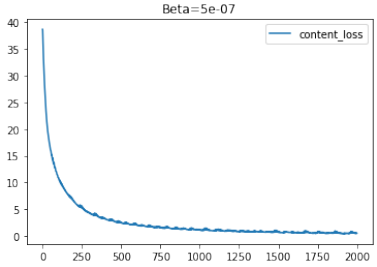
The plots of the loss for $\beta = 5 * 10^{-7}$ shows that it converges pretty quickly around 1000 steps.



(a) Total Loss



(b) Style Loss



(c) Content Loss

Figure 11: Losses plots

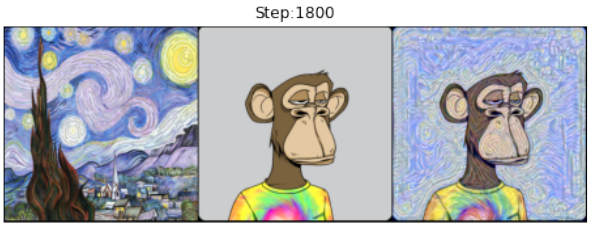
5.6 Results - NST Samples

5.6.1 NST Samples

Here I present certain samples obtained from the model for the hyperparameters chosen. We can clearly see that the ape NFT get styled in the original photo.

(a) Epoch 9

(b) Epoch 13



(c) Epoch 17

(d) Epoch 17

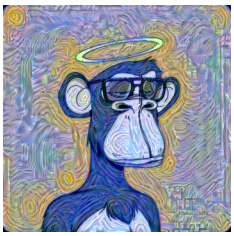
(e) Epoch 19

Figure 12: Generated samples at the middle of training

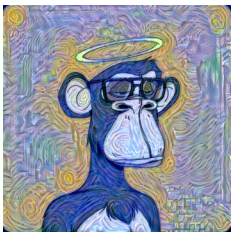
Figure 13: Start of training.

5.6.2 Experience with different Betas

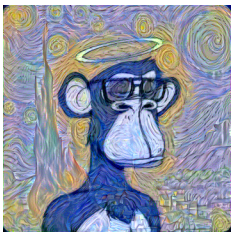
Figure 16 present samples obtained from the model with different β values with a fixed $\alpha = 1$ after running the model for 2000 steps, we can see after certain threshold, here around $\beta = 10^{-5}$ the style of the generated image become strongly influenced by the style image, while for β smaller than 10^{-5} we can barely see the difference between the generated images.



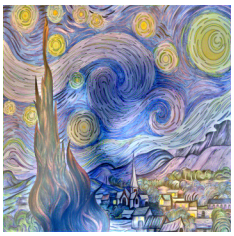
(a) $\beta = 0.1$



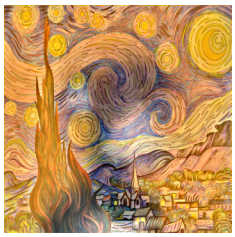
(b) $\beta = 0.003$



(c) $\beta = 10^{-5}$



(d) $\beta = 10^{-7}$



(e) $\beta = 10^{-9}$

Figure 14: Generated samples with different β values

6 Conclusion and Reflection

The aim of this task was to learn how to build and train deep learning models able to produce artistic output. In this task I tried creating AI Art with 2 methods, Deep Convolutional GANs for producing an output that mimics the characteristics and distribution of the dataset, and Neural Style Transfer to borrow the style of an image to create new meaningful images.

The process of both DCGAN and NST is different from the normal procedure followed in machine learning, for example in classification tasks where the model is trained to map an input image to certain class. This shows that there are more creative ways and applications of deep learning than the standard approaches.

The images obtained from DCGANs gave results that showed that the model learned the distribution of the dataset. The output is still not optimal but given the results obtained there is certainly a room for future improvements and more advanced recent models can be experimented with.

The result obtained from NST from the style images shown in the report were satisfactory, however results I obtained from other image were less poignant. Mostly because the style features of these images were not as strong as those in the one presented here. The result from the NST algorithm depend on the style image, and also the style definition, style loss is definitely an area to explore.

7 Code

Github Link

References

[1] Goodfellow, I. et al., 2014. Generative adversarial nets. In Advances in neural information processing systems. pp. 2672–2680.

[2] Radford, A.; Metz, L. and Chintala, S. (2015), 'Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks'

[3] Gatys, L. A.; Ecker, A. S. and Bethge, M. (2015), 'A Neural Algorithm of Artistic Style.', CoRR abs/1508.06576 .

[4] Zeiler, M.D., Fergus, R. (2014). Visualizing and Understanding Convolutional Networks. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds) Computer Vision – ECCV 2014. ECCV 2014.