Coding in AI CPE 393

Project Report

Augustin Morieux, 62540460024

Kunch Ringrod, 5870501105

Objective: *Build a deep learning algorithm that creates art pictures in Basquiat's style*

Abstract: The project is about using Deep Learning techniques to generate art work in the style of painter Jean-Michel Basquiat. The Generative Adversarial Network (GAN) technique allows us to obtain a first set of output with the Basquiat's typical content (choice of colours, distribution of colours). This first output is far from satisfactory because it is limited by the constraints often faced by GANs. The resolution is inadequate, and the Basquiat's style is absent. In order to overcome these constraints, we applied a super resolution algorithm and then created an artistic style transfer algorithm. The result is a new, original piece with satisfactory resolution and that displays the content and style typical of Basquiat's art work. This is a novel approach in Deep Learning applied to paintings. Other approaches mix the work of multiple painters in the dataset and thus don't need to capture the particular style of an artist.

Introduction

The goal of this project is to leverage Jean-Michel Basquiat's art work to create, by means of deep learning, new pieces that would display this painter's typical content and style. To do so, we will use the Generative Adversarial Network technique (GAN). AI engineers have already used GANs in the painting domain. However, these attempts combine multiple painters in the dataset, not one artist's work only. Therefore, their output is impersonal.

What makes Basquiat's work particularly interesting for this project is that this painter died when he was young, at 27 years of age. His production is relatively limited, which raises interesting problems because GANs typically require an abundance of data. Jean-Michel Basquiat was born in Brooklyn (New-York) on December 22nd, 1960 and died in Manhattan (New York) on August 12th, 1988. His father was from Haiti and his mother was Puerto Rican. The combination of his themes (poverty, criticism of colonialism) and pictorial movement (neo-expressionism)

makes his work highly attractive for some collectors. One of his paintings sold for US $ 110.5 million in 2017 at a Sotheby's auction.

Edmond de Belamy is a well-known piece based on the GAN technique (Figure 1). Once printed and brought to a Christie's auction in 2018, it surpassed pre-auction estimates which ranged from US $ 7,000 to 10,000 and was sold for US $ 432,500. This GAN piece was created by the Paris-based arts-collective named *Obvious*. The engineers exploited 15,000 portrait paintings by multiple different painters. This kind of result is technically interesting, yet it is not immune to the general criticism made about AI-based pictorial output: it does not reflect any particular intentionality and its style is impersonal.

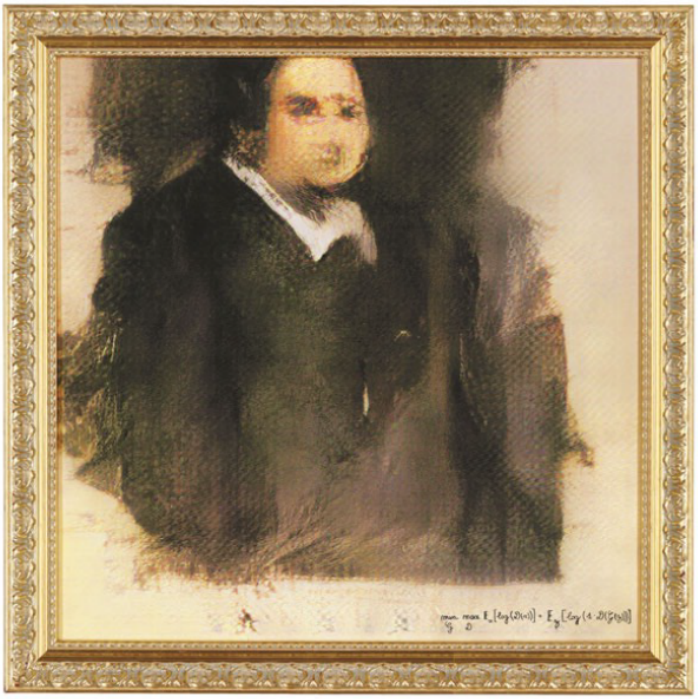

Figure 1. *Edmond de Belamy*, GAN portrait painting, 2018

In order to produce a new piece that is faithful to the artist's style we followed four steps:

1. Data preparation, notably using data augmentation to remedy the relatively limited amount of Basquiat's paintings;
2. Creation of new pieces based on GAN;
3. Optimisation of output by means of super resolution and artistic style transfer algorithms;
4. Selection of the best piece among those obtained, by means of the GAN discriminator.

The next sections of the report describe each of these steps in details. ¶

1. Data preparation

This preparation required data scraping because there is no exiting dataset that contains all Basquiat's work. We then had to resort to data augmentation because of the limited amount of paintings produced by Basquiat. Eventually resizing was necessary.

1.1. Data scraping

We had to create our own dataset, by downloading all Basquiat's canvass paintings available on the WikiArt website [1]. Basquiat created about 200 pieces of art work but only 156 are available on the WikiArt website. Among those, many are not canvass paintings, they range from street tags to drawings on objects such as chairs or helmets. Our first dataset (Data1) contains 76 canvass paintings. Figure 2 shows a sample from Data1. This dataset is about 200 times less rich than the dataset used for the *Edmond de Belamy* output. Such a limited dataset is a problem for deep learning, and we had to tackle this issue.

Figure 2. Pictures from Data1

1.2. Data Augmentation

Because it would be practically impossible to apply deep learning algorithms to 76 images, we had to resort to data augmentation. Using different colours is often a solution in data augmentation, for instance to teach a neural network to recognize cars. However, the choice of colours is usually an intrinsic part of an artist's work and changing the colours would have interfered with the artist's style. Therefore, we could perform data augmentation only by means of some rotations, zooming, and flipping. The code we created to this end is available in our "Data_Augmentation.py" file. By applying those techniques, we obtained 532 images (Figure 3). The new dataset (Data2) is much more appropriate in the context of deep learning.

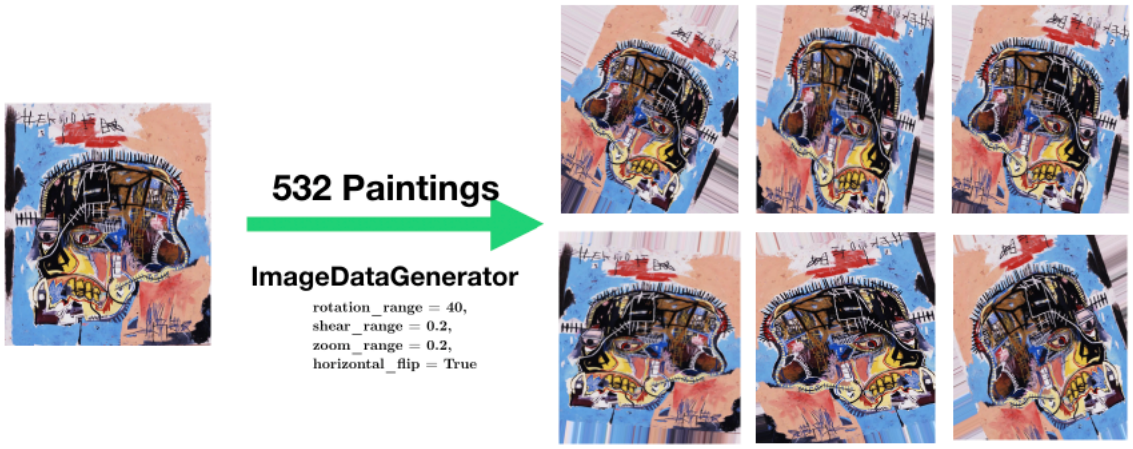

Figure 3. Data augmentation: pictures generated from one picture

1.3. Data transformation

After performing data augmentation, we had to resize all the images. We opted for the conventional resizing: width of 500 px and height of 420 px. As a result of this data transformation we obtained a new dataset, much more amenable to deep learning (Data3). The next section describes how we built the generative model.

2. Creation of new pieces based on GAN

Now that we have a proper dataset it is possible to proceed with the creation of the generative model. GAN is an effective technique to create a generative model.

2.1. Summary description of GANs

GANs are a technique for teaching a deep learning model to capture the distribution of training data so that it becomes possible to generate new data from that same distribution [2]. A GAN consists of two distinct models: a *generator* and a *discriminator*. The generator's function is to generate fake images that look like the training images. The discriminator's function is to look at an image and determine whether it is a real training image or a fake image from the generator. In the training phase, the generator tries to beat the discriminator by generating better and better fakes, while the discriminator learns to classify the real and fake images. Equilibrium is reached when (i) the generator produces perfect fakes that seem to originate in the training data, and (ii) the discriminator guesses at 50% confidence that the generator output is real or fake. A typical GAN architecture is illustrated in Figure 4.

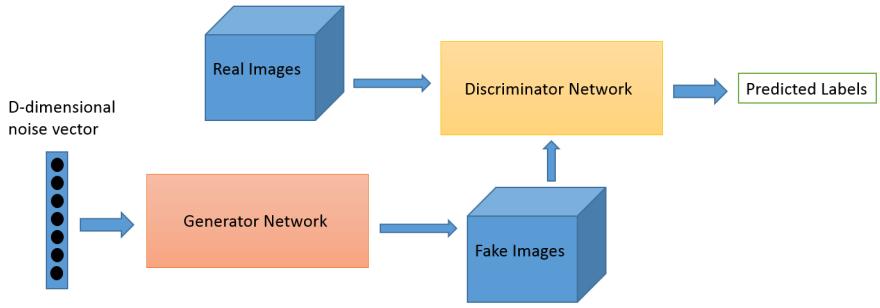

Figure 4. A GAN architecture [3]

2.2. Our GAN

Our GAN is based on the typical GAN architecture. Details about the main components of our GAN are shown in Figure 5.

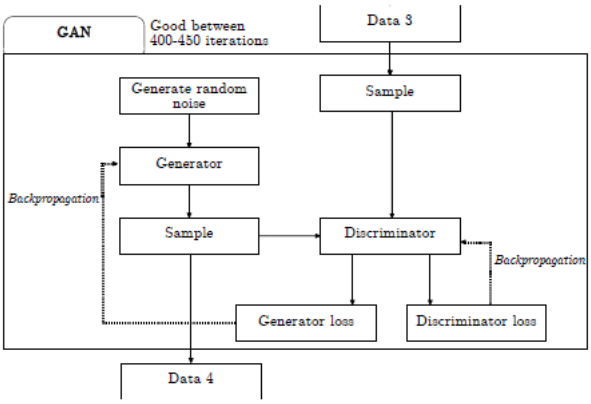

Figure 5. Our GAN architecture

Our generator and discriminator models are displayed in Exhibits 1 and 2 at the end of the report. Our generator takes random noise as an input, and then generates a sample of four images at each iteration (Figure 6). The discriminator, fed by the 532 resized images of Data3, then analyses the four images and classifies them as fake or real. Two categories of losses are then calculated: the generator loss, and the discriminator loss. Gradients are then obtained by backpropagation of both the generator loss and discriminator loss. The gradients are then used to update the weights in the generator and discriminator.

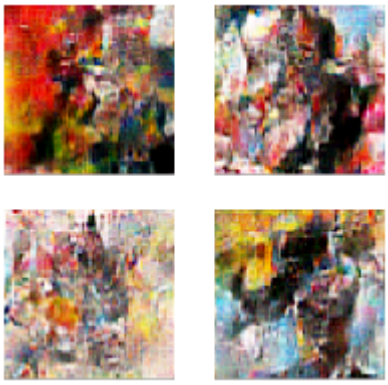

Figure 6. Output of an iteration

2.3. Results

The optimal output is produced between 400 and 450 iterations. We obtained a new dataset, consisting of four outputs for each iteration between the 400th and 450th iterations, i.e. 200 images (Data4). Colours are not satisfactory below 400 iterations. In particular, below this threshold, the output does not display the figurative darker pattern typical of Basquiat's content. Beyond 450 iterations, the Basquiat's typical figurative darker pattern disappears. Figure 7 shows how the loss evolves as a function of the number of iterations. The discriminator loss is in blue and the generator loss is in orange. The bottom of Figure 7 shows five sets of outputs: the first set of four images happens between 0 and 100 iterations, the second set is between 100 and 200, the third is between 200 and 300, the fourth is between 300 and 400, and the fifth is between 400 and 450.

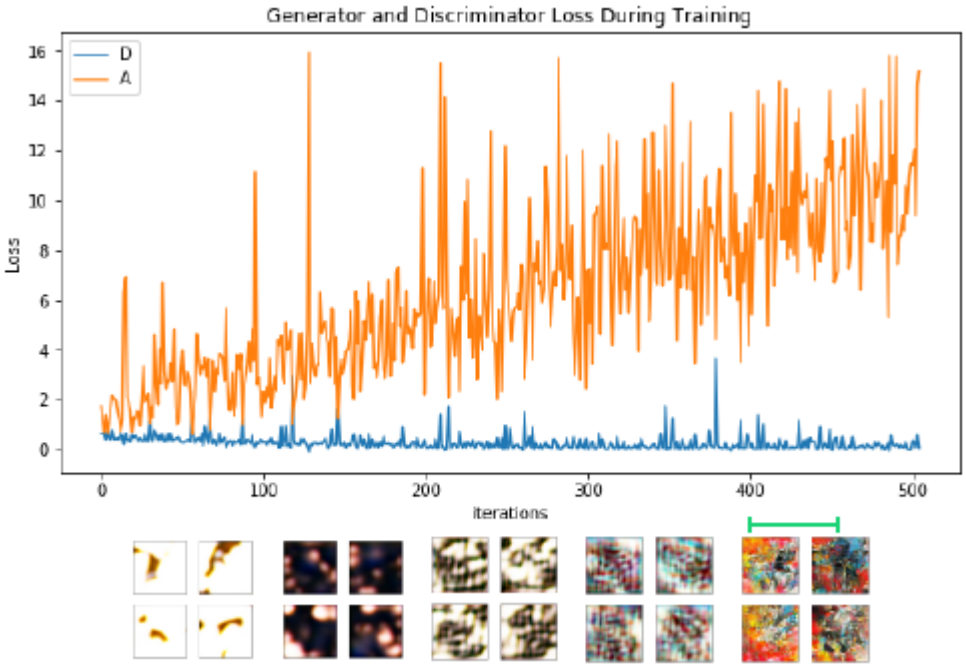

Figure 7. Evolution of output according to the number of iterations

An illustration of output which is satisfactory in terms of content is displayed in Figure 8. This piece is obtained at the 427th iteration. It is more satisfactory in terms of choice and distribution of colours than images below the 400th and beyond the 450th iterations. Figure 8 clearly shows the figurative darker pattern typical of Basquiat's work.

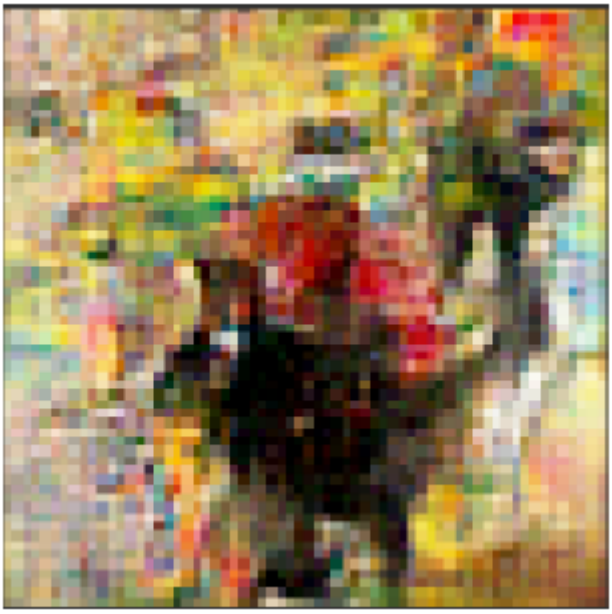

Figure 8. One of the four outputs produced at the 427$^{\text{th}}$ iteration

Content is about the choice and distribution of colours. However, two other attributes must also be taken into account: resolution and style. The piece represented in Figure 8 is very fuzzy. Moreover, it does not properly reflect Basquiat's style, notably the calligraphic symbols that often show in Basquiat's art work. The way we addressed these two shortcomings is described in the next section.

3. Optimization of output in terms of resolution and style

The resolution problem is a usual one when using GANs. We tackled it by using a super resolution network. The style problem is more specific to our objective and required another type of solution.

3.1. Modify the resolution to improve the visual aspect

Three engineers from the Max Planck Institute for Intelligent Systems have developed a model for super resolution through automated texture synthesis. The model is called *EnhanceNet* [4]. The model uses feed-forward fully convolutional neural networks (CNNs) with an adversarial training setting. This allows to significantly increase the quality of an image at high magnification ratios. Figure 9 describes the architecture of the CNN of *EnhanceNet*.

Output size	Layer
$w \times h \times c$	Input I_{LR}
$w \times h \times 64$	Conv, ReLU
	Residual: Conv, ReLU, Conv
	...
$2w \times 2h \times 64$	2x nearest neighbor upsampling Conv, ReLU
$4w \times 4h \times 64$	2x nearest neighbor upsampling Conv, ReLU
	Conv, ReLU
$4w \times 4h \times c$	Conv
	Residual image I_{res}
	Output $I_{\text{est}} = I_{\text{bicubic}} + I_{\text{res}}$

Figure 9. Architecture of the CNN of *EnhanceNet*

By applying *EnhanceNet* 11 times to each of the 200 images of Data4, we obtained a new set of 200 images with much better resolution (Data5). Figure 10 displays one image in Data5. This image is the result obtained after applying 11 times *EnhanceNet* pre-trained model to the image of Figure 8.

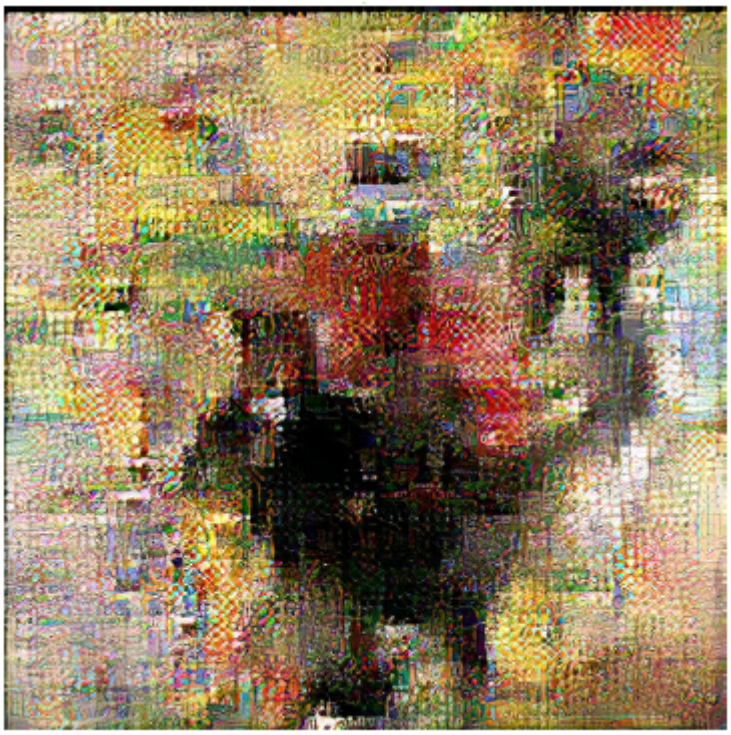

Figure 10. Illustration of output with super resolution

This output is obviously still satisfactory in terms of content (choice and distribution of colours) and is now much better in terms of resolution. The next step is to find a solution to better reflect Basquiat's style.

3.2. Artistic Style Transfer

The content of Data5 does not fully reflect Basquiat's style. As noted earlier, the calligraphic symbols typical of Basquiat's art work do not show up in the output. We used artistic style transfer to solve this problem. In theory the ideal approach would have been to extract Basquiat's style from the 76 original paintings of Data1 and then apply it to the 200 images of Data5. However, this would have taken 2,500 hours because of the computational power of our machines. We had to find a shortcut. In Data1, i.e. in the set of 76 original paintings, we selected four pieces that were particularly emblematic of Basquiat's style. We randomly selected five images in Data5. The expected time to achieve the necessary calculation went down to three hours.

In order to perform the artistic style transfer from the four emblematic pieces of D1 to the five randomly selected images of D5, we used the pre-trained VGG-16 convolutional neural network. The approach is based on a 2015 paper [5]. The way we implemented the model is described in Figure 11.

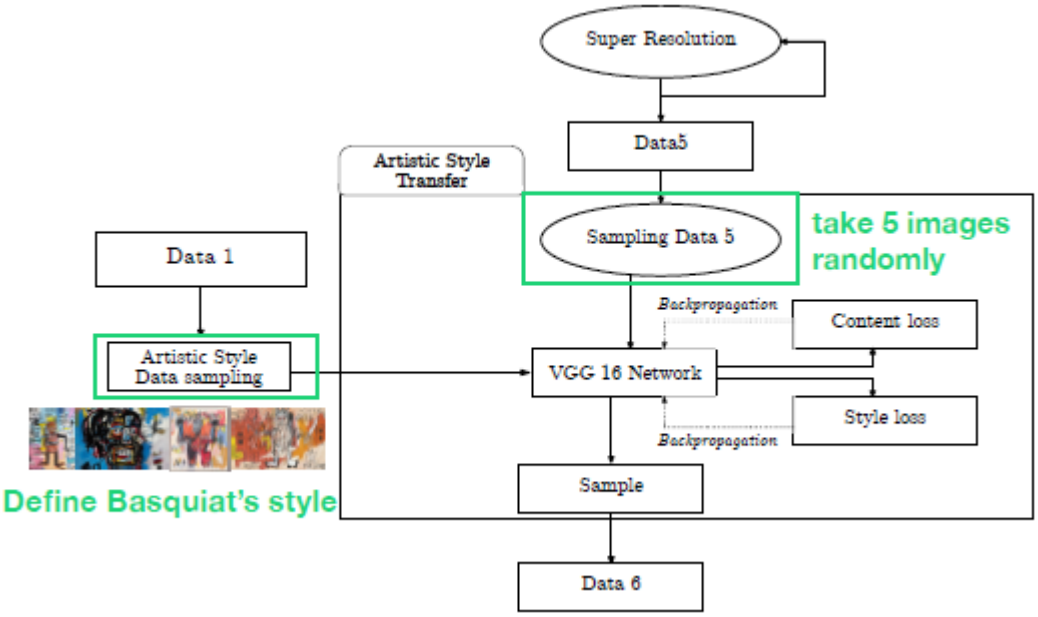

Figure 11. Architecture of our artistic style transfer model

The result is 20 images (Data6) that now much better reflect Basquiat's style, and that still display adequate content and resolution as these two attributes were not modified by the artistic style transfer. Figure 12 shows an output after artistic style transfer.

Figure 12. An output in Data6

Among the 20 images of Data6, some may yield a much better fit with Basquiat's content and style than others. The next step is to find a way to select the image with the best fit. ¶

4. Choose the best image

We took Data6 as an input for our GAN discriminator that was trained in step 2. For each image of Data6 we obtained a generator loss from the discriminator. Based on the generator loss we obtained a hierarchy of images. Figure 13 shows the architecture of our model for choosing the best picture, i.e. the image at the top of the hierarchy.

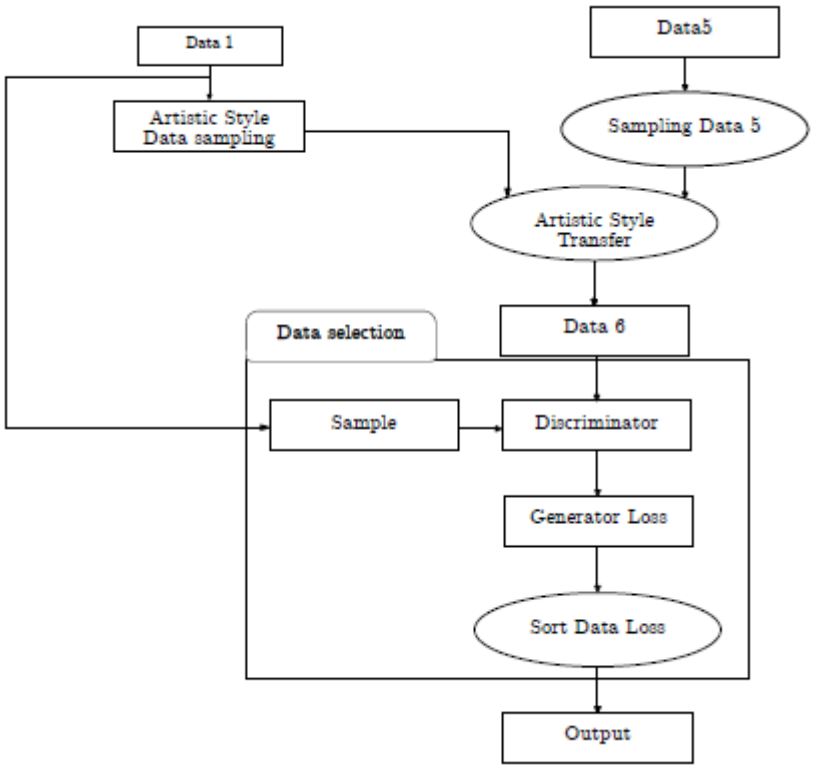

Figure 13. Architecture to select the best image

The image that was selected by means of this process, among the 20 images of Data6, is displayed in Figure 14. This new, original piece displays Basquiat's typical content: choice of colours and distribution of colours (notably the darker pattern). Second, it has a satisfactory resolution. Third, it captures Basquiat's style, notably the calligraphic symbols that often show in Basquiat's art work.

The general architecture can be found in appendix 3.

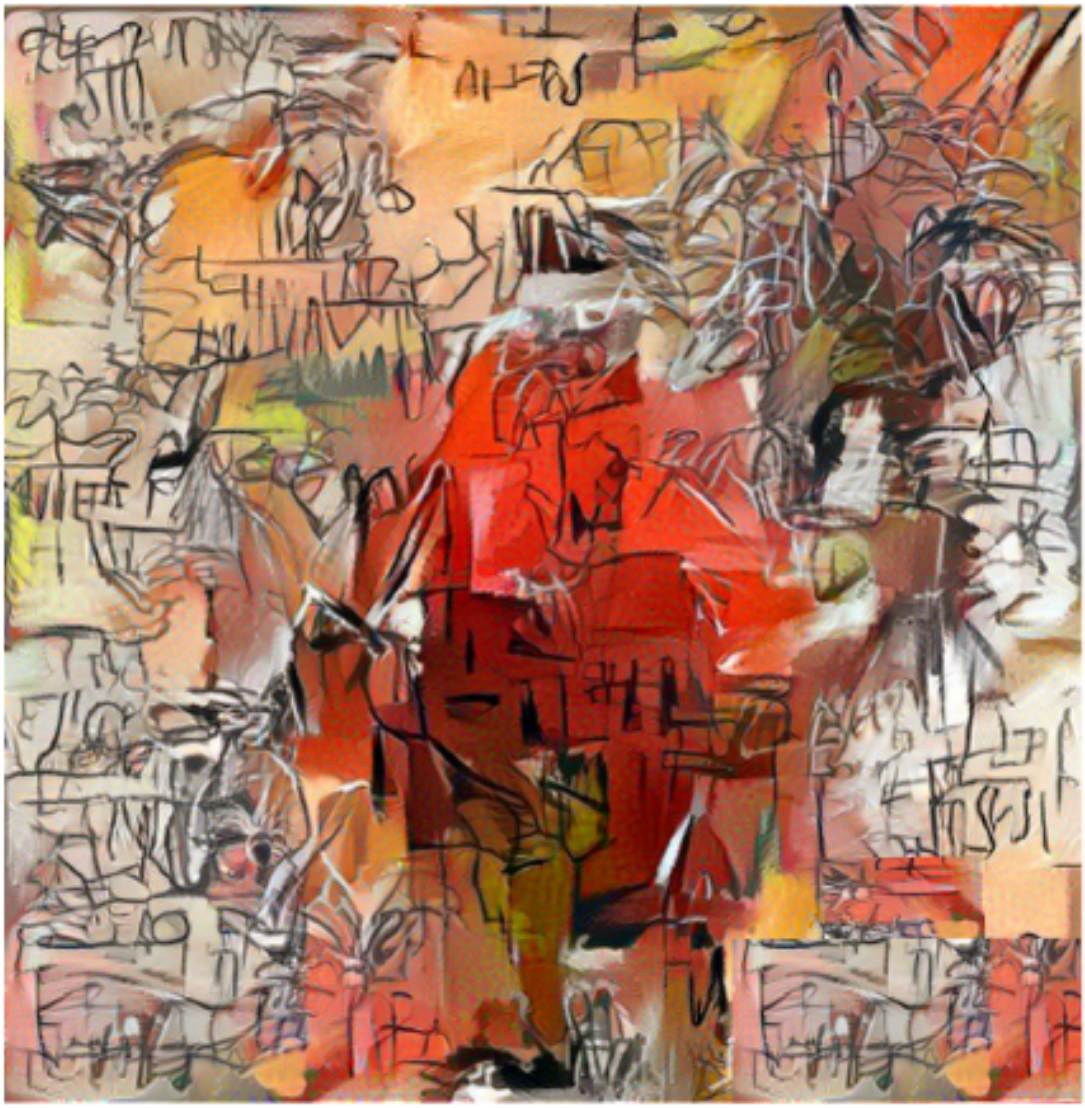

Figure 14. Best image selected by our model

Conclusion

This project is original because, unlike many GAN-based attempts to create new art pieces, we focused on a single painter. As a result, we needed to identify and learn ways to perform deep learning on a relatively small dataset consisting of 76 paintings.

The final output is satisfactory in terms of content, resolution and style. This satisfactory output was made possible by applying super resolution and artistic style transfer neural networks to the output of our GAN's generator. This boosted our GAN's generator.

Looking forward, the two main areas for improvement relate to the discriminator and computational power. With more time, the discriminator could be better trained. With greater computational power, it would be possible to produce more images before selecting the best one.

References

[1] WikiArt website (last accessed November 25th, 2019): https://www.wikiart.org/en/jean-michel-basquiat

[2] GANs were invented in 2014 and first described in a paper written by researchers at Université de Montréal (last accessed November 25th, 2019): https://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf

[3] Source (last accessed November 25th, 2019): https://skymind.ai/images/wiki/gan_schema.png

[4] Description of the approach developed by the three Max-Planck engineers for super resolution ("EnhanceNet" approach, last accessed November 25th, 2019): https://webdav.tuebingen.mpg.de/pixel/enhancenet/files/EnhanceNet_paper.pdf

[5] Article that lays out the artistic style transfer solution we used (last accessed November 25th, 2019): https://arxiv.org/pdf/1508.06576.pdf

Appendices

Appendix 1. Generator

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, 100)	0
dense_1 (Dense)	(None, 131072)	13238272
leaky_re_lu_1 (LeakyReLU)	(None, 131072)	0
reshape_1 (Reshape)	(None, 32, 32, 128)	0
conv2d_1 (Conv2D)	(None, 32, 32, 256)	819456
leaky_re_lu_2 (LeakyReLU)	(None, 32, 32, 256)	0
conv2d_transpose_1 (Conv2DTr	(None, 64, 64, 256)	1048832
leaky_re_lu_3 (LeakyReLU)	(None, 64, 64, 256)	0
conv2d_2 (Conv2D)	(None, 64, 64, 256)	1638656
leaky_re_lu_4 (LeakyReLU)	(None, 64, 64, 256)	0
conv2d_3 (Conv2D)	(None, 64, 64, 256)	1638656
leaky_re_lu_5 (LeakyReLU)	(None, 64, 64, 256)	0
conv2d_4 (Conv2D)	(None, 64, 64, 3)	37635

Total params: 18,421,507
Trainable params: 18,421,507
Non-trainable params: 0

Appendix 2. Discriminator

Layer (type)	Output Shape	Param #
input_2 (InputLayer)	(None, 64, 64, 3)	0
conv2d_5 (Conv2D)	(None, 62, 62, 128)	3584
leaky_re_lu_6 (LeakyReLU)	(None, 62, 62, 128)	0
conv2d_6 (Conv2D)	(None, 30, 30, 128)	262272
leaky_re_lu_7 (LeakyReLU)	(None, 30, 30, 128)	0
conv2d_7 (Conv2D)	(None, 14, 14, 128)	262272
leaky_re_lu_8 (LeakyReLU)	(None, 14, 14, 128)	0
conv2d_8 (Conv2D)	(None, 6, 6, 128)	262272
leaky_re_lu_9 (LeakyReLU)	(None, 6, 6, 128)	0
flatten_1 (Flatten)	(None, 4608)	0
dropout_1 (Dropout)	(None, 4608)	0
dense_2 (Dense)	(None, 1)	4609

Total params: 795,009
Trainable params: 795,009
Non-trainable params: 0

Appendix 3. The general architecture