GANs learning to draw oil paintings

Beyrem Kaddech

Bachelor's thesis

GANs learning to draw oil paintings

Beyrem Kaddech

29. August 2019

Institute for Data Processing
Technische Universität München

Beyrem Kaddech. *GANs learning to draw oil paintings.* Bachelor's thesis, Technische Universität München, Munich, Germany, 2019.

Supervised by Prof. Dr.-Ing. K. Diepold and Mathhias Kissel; submitted on 29. August 2019 to the Department of Electrical Engineering and Information Technology of the Technische Universität München.

Institute for Data Processing, Technische Universität München, 80290 München, Germany, `http://www.ldv.ei.tum.de/`.

Abstract

Generative Adverserial Networks (GANs) [1] have established themselves in the last couple of years as a staple in deep generative models. They provide state of the art results and inception scores [2] on many of the well studied and experimented on datasets such as Cifar100 in and LSUN-Bedrooms in [3]. In Context of this Bachelor thesis I implement the Wasserstein GAN model [4] on a new dataset that I extract from a hand painted film. I test the robustness of my network to gain further insight into the necessary requirements for not only the stability of the model but also good quality generated images. Through trial and error I establish best practices relating to the choice of hyperparameters and the degree of freedom the model allows. I also adopt a hierarchical approach where I create different subsets of our training data and compare the generated images from each subset. This allows me to draw conclusions about which training data delivers the best performance, assumptions about the behaviour of the model and ideas for future research.

Contents

1 Introduction

Background

In the wake of the relative success of deep discrimination models in supervised as well as unsupervised learning in comparison to the generative models, Generative Adversarial Networks (GANs) were developed as a way to make use of the back-propagation algorithms with well-behaved gradients in discrimination models and apply them in generative tasks [1].

The model consists of two neural networks pinned against each other in a minimax two-player game. One neural network, the discriminator, tries to distinguish between the real data and the generated one, that is to say: to correctly label each one and thus minimize its cost function. The other neural network, the generator, gradually generates data that best represents the real data distribution and by that rendering it progressively more difficult for the discriminator to distinguish between the two, thus maximizing the discriminator's cost function [1]. In this back and forth "competition" both networks get better at their respective tasks, and as the training progresses, better outputs (more representative of the striven for data distribution) are generated.

This first GAN has proven to be very inspiring for further research. Since then, several iterations have been published and discussed that build on the basic model. The most notable milestone is the introduction of the Wasserstein GAN (WGAN) [4] which is the go-to base model in our work.

In practice, impressive results have been achieved in previous works with WGANs and others, especially in image generation tasks [5, 6, 7]. Still, these remain dependant on the choice of training data as different sets behave very differently and in some cases, unpredictably [1]. Furthermore, although some progress has been made [8], an underlying theoretical understanding of the stability of GANs is still an open research area. As a result of this, as is common practice in deep learning models, the choice of the architecture and the different parameters that describe the network are not only remarkably fine-tuned but also came about through an extensive trial and error approach [1]. The specifics of why individual values and architectures converge or deliver way better results than others remains mostly a speculative matter. Under the absence of a mathematical explanation, a different approach seems to be to develop best practices while dealing with these models and empirical observations that could be insightful in developing new ideas and techniques.

Motivation

The lack of concise theoretical grounds motivates our effort to further experiment with the model and its compatibility with a new and somewhat different dataset. This new dataset consists of the different frames I extract from the film "Loving Vincent" [9]. The film was especially interesting to us as it consists of hand-painted oil paintings that mimic the style famously used by "Vincent Van Gogh" and in some cases some of his specific paintings. The sheer number of paintings (approximately 68000) and their stylistic resemblance to each other inspired their use as a training set. Besides, the prospect of producing highly artistic work using deep learning was especially inspiring.

Goals

One of the main goals of the thesis is to create a GAN model that is well suited for our "Van Gogh dataset". Through this, I gain further insight into the world of GANs and develop a common sense understanding of best practices on how to use the different parameters and algorithms and, in some cases, how to adjust them to the different needs. In a parallel fashion, I aim to find a subset of the data that is best suited for my model and compare the different subsets, primarily in terms of quality and generalization. On this basis, I propose some assumptions on which datasets behave best with a given model.

Approach

I divide the thesis into 3 main sections:

- First, I thoroughly describe the model as well as the different methods used throughout. The architecture of both generator and discriminator/critic (in this case more accurately critic) is detailed and the training algorithm is explained. I also justify the choice of the cost and optimization functions, the different parameters and the used techniques.

- Then I lay out the road map of the general process and list the different experiments I performed including extraction and preprocessing of the data. The methodology used in these experiments and the motivation behind it is also discussed. I share my results for different iterations of the model and different subsets of data.

- Lastly, I leverage the results I obtained in the preceding chapter to analyze the behaviour of the GAN. The interpretation of some curious observations during the experiments allows me to put forth some assumptions that could motivate future work. Also, I list the best practices that helped me achieve the most impressive results and discuss those to the limit of my understanding. Finally, I list areas where my approach was lacking or erroneous.

2 Theoretical Model and Methods

2.1 Architecture

Different implementations of GAN models that satisfy different needs require a diverse choice of Neural Network. In the context of image generation, both my generator and discriminator are implemented as deep Convolutional Neural Networks(CNNs) [10] as these have proven to be one of the best current models for any image processing.

Black box model

First, to understand the general function of both generator and discriminator, I utilize a black box representation as is illustrated in figures 2.1 and 2.2.

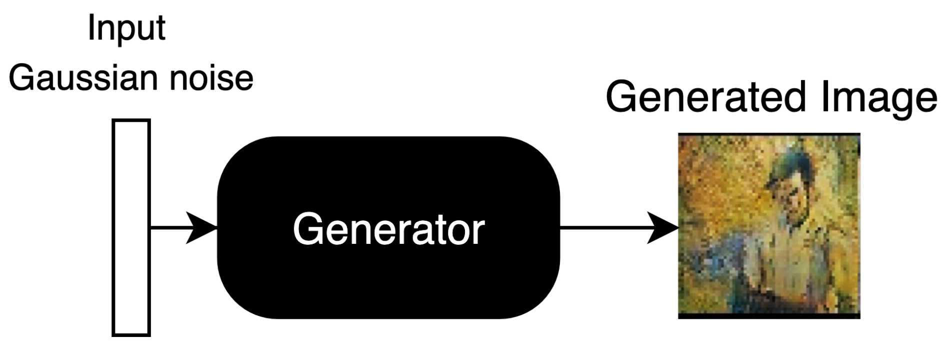

Figure 2.1: The generator as a black box model

A good Generator is one that is able to function, after training, independently from the discriminator. That allows us to implement it in different applications without having to worry about any dependencies. Most applications rely on the generator and less so on the discriminator. Thus, training GANs in most cases is about training a generator that can produce good images

Its input is a random noise from a normal distribution of size 128. The output is an image of size 64x64x3.

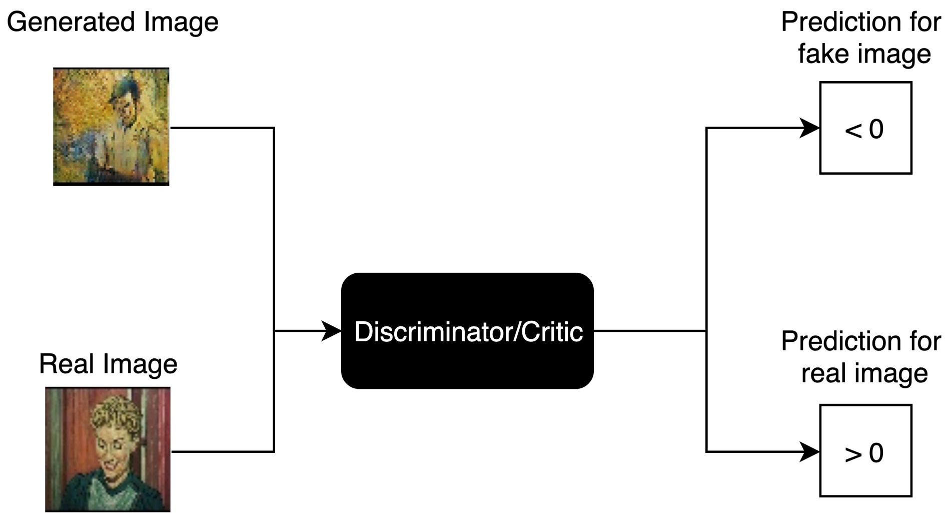

Figure 2.2: The critic as a black box model

The discriminator is a CNN in a more usual sense. It performs the task of trying to "label" the input images. It decides whether they are fake, which means that they were created by the generator, or real, which means that they come from my dataset. In the more traditional GANs, this is done by giving a probability between 0 and 1 of the image being real. In the more recent Wasserstein GAN (WGAN) [4] which is the base of my implementation here, the output reflects how real the input image is. This approach is adopted from the "value function" in reinforcement learning. To better reflect its new function, we rename the discriminator to the Critic in a similar fashion to an art critic that can judge how well a painting reflects a specific style. The input of the Critic is either a generated image or one from my dataset. It outputs a single value with highly positive values indicating that the image is indeed real and negative ones showing that the image is fake.

To better understand the inner workings of both CNNs, I now discuss their composition and the exact architecture I ended up using in my implementation. I mainly referred to the DCGAN paper [11] with regards to my choices as have most of the recent works with GANs have. The motivation is the success of DCGAN and the scale that the DCGAN implementation allows me to work with in terms of desired resolution and the number of layers. This choice enables me to perform my experiments in a realistic time frame for a bachelor thesis and is adequate for the available hardware.

2.1.1 Generator

Figure 2.3 shows the architecture of the Generator.

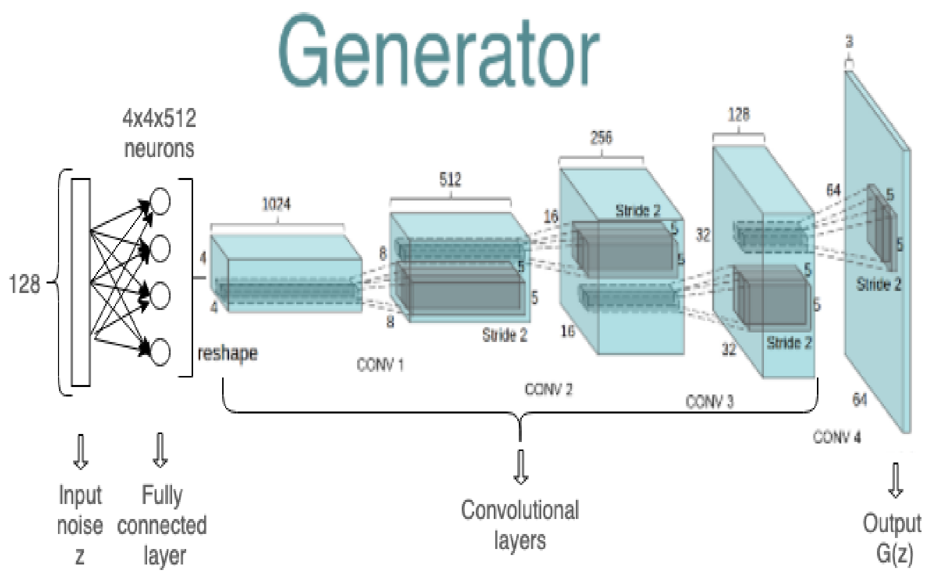

Figure 2.3: The generator as a Convolutional Neural Network [11]

The generator consists of 5 layers. A summarization of the shape of each layer and the filters used follows:

- Layer 1: Fully connected layer with 4x4x512 neurons
- Layer 2: 4x4x1024 with 512 3x3 Filters
- Layer 3: 8x8x512 with 256 3x3 Filters
- Layer 4: 16x16x256 with 128 3x3 Filters
- Layer 5: 32x32x128 with 64 3x3 Filters
- Output : 64x64x3 Output

A couple of things to note here are:

- **The lack of pooling layers:** I don't utilize pooling layers in my implementation and opt for strided convolutions as they are, in our case, more computationally efficient. For each layer, it significantly reduces the amount of computation that has to be done by the network in the subsequent layers. It compresses multiple convolutions. At the same time, this layer applies a stride of 2 that downsamples the image. Because each layer does convolution and downsampling at the same time, the operation becomes significantly cheaper computationally. If you use stride 1 and pooling for downsampling, then you end up with convolution that does more computation plus extra computation for the next pooling layer. The same trick was used in ResNet [12], SqueezeNet[13] and some other neural network architectures.

- **Residual Block:** I also implement a residual block [12] where each layer doesn't only feed to the next layer but also the one after it. Networks generally improve accuracy with added layers which motivates the deep network approach. However, it has been proven that due to some problems like vanishing gradients if we have sufficiently deep networks, it may not be able to learn more straightforward functions like the identity function. Residual block presents itself as a solution that allows you to skip the training of few layers using skip-connections or residual connections. Because of these skip connections, we can propagate more significant gradients to initial layers, and these layers also could learn as fast as the last layers, giving us the ability to train deeper networks [12].

2.1.2 Critic

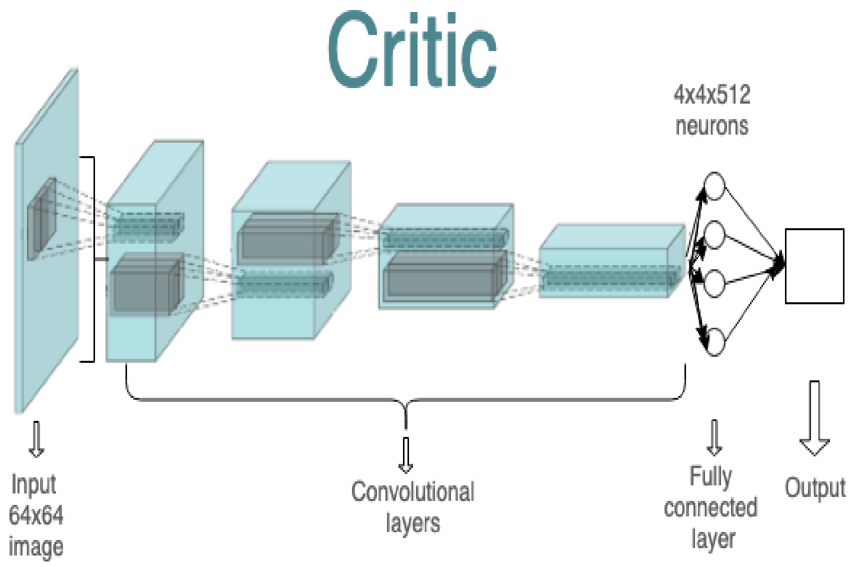

Figure 2.4: The critic as a Convolutional Neural Network [11]

Figure 2.4 shows the architecture of the Critic.
The architecture of the Critic is highly similar to that of the generator :

- Layer 1: 32x32x64 with 128 3x3 filters
- Layer 2: 16x16x128 with 256 3x3 filters
- Layer 3: 8x8x256 with 512 3x3 filters
- Layer 4: 4x4x512 with 512 3x3 filters
- Layer 5: fully connected layer with 4x4x512 neurons
- Output: a single output.

In a similar fashion to the generator, I use strided convolutions instead of pooling operations and utilize residual block between the layers.

2.2 Methods and Algorithms

I make use of the architecture, as mentioned earlier, to perform computations on my data while utilizing standard and modern techniques. I focus on such algorithms and methods in this section.

Figure 2.5 helps summarize the flow of information within my network and how I utilize the outputs of my generator and discriminator model within my implementation.

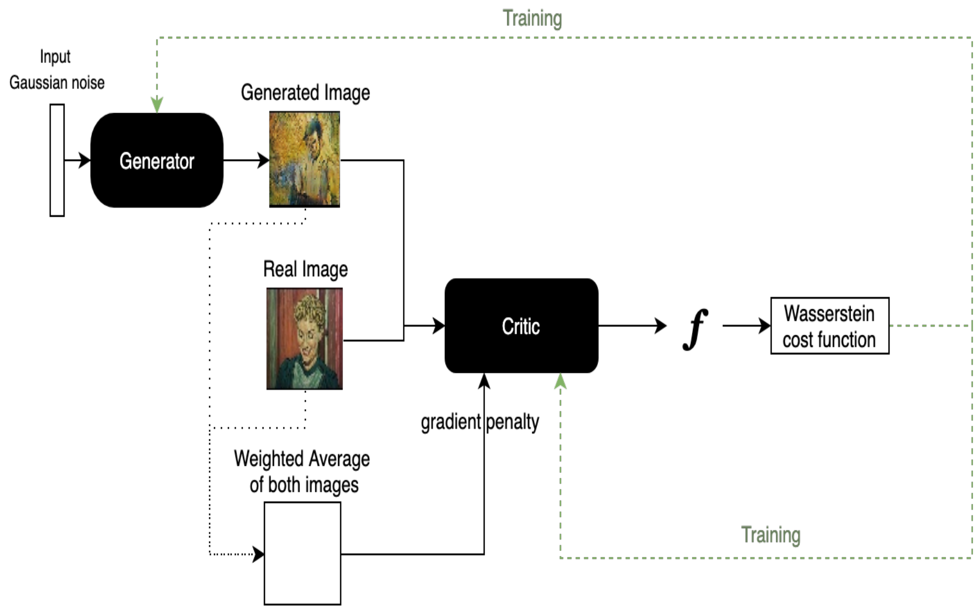

Figure 2.5: Model representation

Cost function: As a cost function, I implement the Wasserstein cost function introduced in [4] and improved upon in [3]. The Wasserstein cost function is based on the Earth Mover distance [14] between two data distributions p and q, which is the minimum cost of converting a data distribution q to a data distribution p. Wasserstein GAN (WGAN) proposes a new cost function using the Earth Mover distance that has a smoother gradient everywhere. WGAN learns whether the generator is performing or not. The difference between WGAN gradients and traditional GAN is shown in the following table from [4]

	Discriminator/Critic	**Generator**
GAN	$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right]$	$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)$
WGAN	$\nabla_w \frac{1}{m} \sum_{i=1}^{m} \left[f(x^{(i)}) - f(G(z^{(i)})) \right]$	$\nabla_\theta \frac{1}{m} \sum_{i=1}^{m} f(G(z^{(i)}))$

Figure 2.6: Gradients comparison in GANs and WGANs [4]

With x being the real data distribution, z the input noise, D the discriminator, G the generator and f a 1-Lipschitz function [4]. So to calculate the Wasserstein distance, we need to find a 1-Lipschitz function. A differentiable function f is 1-Lipschitz if and only if it has gradients with norm at most 1 everywhere. Like other deep learning problem, we can build a deep network to learn it. That network would be similar to the Critic but outputs a scalar score rather than a probability. We can interpret this score as to how real the input images are.

While finding f we need to enforce the 1-Lipschitz constraint of having gradients with norm smaller than 1. We do this by implementing a gradient penalty [3] which penalizes the model if the gradient norm moves away from its target norm value 1. The critic loss can be expressed as follows:

$$L = \underbrace{\underset{\tilde{\boldsymbol{x}} \sim \mathbb{P}_g}{\mathbb{E}} [D(\tilde{\boldsymbol{x}})] - \underset{\boldsymbol{x} \sim \mathbb{P}_r}{\mathbb{E}} [D(\boldsymbol{x})]}_{\text{Original critic loss}} + \underbrace{\lambda \underset{\hat{\boldsymbol{x}} \sim \mathbb{P}_{\hat{\boldsymbol{x}}}}{\mathbb{E}} \left[(\|\nabla_{\hat{\boldsymbol{x}}} D(\hat{\boldsymbol{x}})\|_2 - 1)^2 \right]}_{\text{Our gradient penalty}}.$$

where $\hat{\boldsymbol{x}}$ *sampled from* $\tilde{x}$ *and* x *with t uniformly sampled between 0 and 1*

$$\hat{\boldsymbol{x}} = t\tilde{\boldsymbol{x}} + (1-t)\boldsymbol{x} \text{ with } 0 \le t \le 1$$

Figure 2.7: Critic loss [3]

Figure 2.8 shows the pseudo-code that summarizes how the gradient penalty is computed and how the training takes place. For my optimization, I use Adam.

```
Require: The gradient penalty coefficient $\lambda$, the number of critic iterations per generator iteration
    $n_{\text{critic}}$, the batch size $m$, Adam hyperparameters $\alpha, \beta_1, \beta_2$.
Require: initial critic parameters $w_0$, initial generator parameters $\theta_0$.
 1: while $\theta$ has not converged do
 2:     for $t = 1, ..., n_{\text{critic}}$ do
 3:         for $i = 1, ..., m$ do
 4:             Sample real data $\boldsymbol{x} \sim \mathbb{P}_r$, latent variable $\boldsymbol{z} \sim p(\boldsymbol{z})$, a random number $\epsilon \sim U[0, 1]$.
 5:             $\tilde{\boldsymbol{x}} \leftarrow G_\theta(\boldsymbol{z})$
 6:             $\hat{\boldsymbol{x}} \leftarrow \epsilon \boldsymbol{x} + (1 - \epsilon)\tilde{\boldsymbol{x}}$
 7:             $L^{(i)} \leftarrow D_w(\tilde{\boldsymbol{x}}) - D_w(\boldsymbol{x}) + \lambda(\|\nabla_{\hat{\boldsymbol{x}}} D_w(\hat{\boldsymbol{x}})\|_2 - 1)^2$
 8:         end for
 9:         $w \leftarrow \text{Adam}(\nabla_w \frac{1}{m} \sum_{i=1}^{m} L^{(i)}, w, \alpha, \beta_1, \beta_2)$
10:     end for
11:     Sample a batch of latent variables $\{\boldsymbol{z}^{(i)}\}_{i=1}^{m} \sim p(\boldsymbol{z})$.
12:     $\theta \leftarrow \text{Adam}(\nabla_\theta \frac{1}{m} \sum_{i=1}^{m} -D_w(G_\theta(\boldsymbol{z})), \theta, \alpha, \beta_1, \beta_2)$
13: end while
```

Figure 2.8: Algorithm of Wasserstein generative adversarial network. [3]

My choice for the hyperparameters:

- α = 0.0001
- β1 = 0
- β2 = 0.9
- Ncritic = 5
- λ = 10.

These were empirically proven to yield good results[1, 3].

One thing to note here is that for every five critic iterations, there is only one generator iteration.

As usual with the newest neural networks, I use batch normalization which computes the mean and variance of each mini-batch and normalizes each feature according to the mini-batch statistics and by doing that forces the input of every layer to have approximately the same distribution in every training step [15]. This technique dramatically improves performance and training speed because each layer must not learn to adapt themselves to a new distribution in every training step.

Note that the improved WGAN paper [3] suggests that batch normalization should not be used in the Critic as it creates a correlation between samples in the same batch, which impacts the effectiveness of the gradient penalty. For this, I implement a layer normalization introduced in [16] for the Critic. It normalizes the inputs across their different features instead of normalizing across the different samples of a batch. This method avoids the problem of creating a correlation between different samples.

For my activation functions, I use the Rectified Linear Unit (ReLU) in all layers except the last layer of the generator where the hyperbolic tangent (tanh) is used.

3 Experiment and Results

Figure 3.1 illustrates my methodology while approaching the task at hand. It consists of 3 main components:

- Preprocessing and data extraction
- The building of the model and training
- Image generation.

These are discussed individually in the next 3 sections.

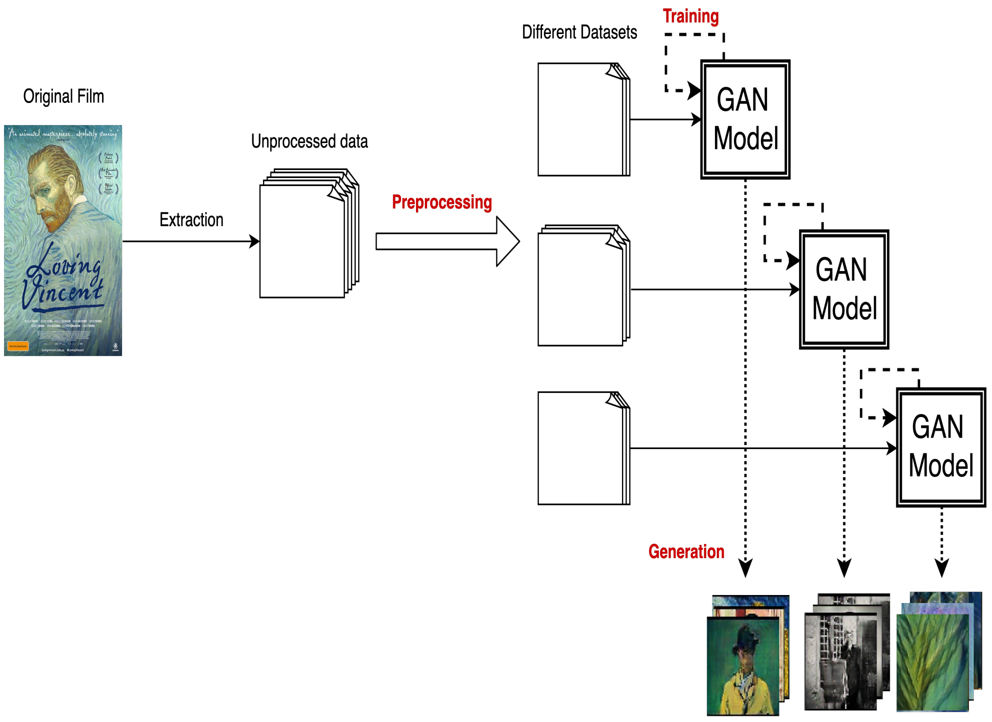

Figure 3.1: Data processing flowchart

3.1 Data extraction and Preprocessing

Data extraction

My starting point was the award-winning film, "Loving Vincent" [9]. It is a homage to the famous painter Vincent Van Gogh. The creators took inspiration from his style of painting and tried to use some of his most famous paintings to create a fictional story where the main characters are some of the subjects of his paintings. The film displays a variety of places, landscapes and characters all painted in the very distinct Van Gogh style. The film consists of 68000 unique oil paintings.

Motivated by the wealth of data at my disposal and their similarity in style, I created a dataset out of the film. For this, I started by extracting frames from the movie. I saved every 3rd frame and seeing that the movie was at 12 frames per second, I were able to obtain 4 images per second for a total of 45000 images. I did this because my initial approach of extracting every single frame produced a dataset with many almost identical images. This is because the film used minor software altercations to create the illusion of movement and interpolate between different paintings, thus raising the frame-rate without having to produce more paintings. A sample of extracted frames from the film (our ground truth) at 64x64 resolution is shown in Figure 3.2

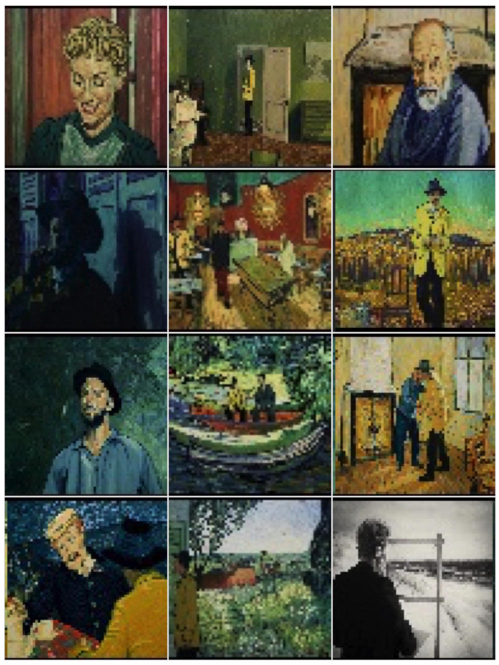

Figure 3.2: Samples from the data extracted from the film

Preprocessing

One goal of this bachelor thesis was to examine the behaviour of the GAN on different hierarchies of a dataset. After close examination of the images and as a result of some of the experiments that will later be discussed, I opted for the following partitioning:

- **General Dataset:** Where I kept all the data extracted apart from a few frames where the screen was black or where there was only text, as these can throw off my training and contribute to undesired results.
- **Black and White Dataset:** The film is split into two parts that cut between one another: a flashback section which is entirely in black and white and presents a slightly different style to and more detail than the other coloured part. The black and white images make about 35% of the total data with approximately 16000 frames.
- **Portrait Dataset:** From the remaining 65% of coloured frames, I extract the images with at least one character in them with their face visible. As the film is dialogue-heavy, I obtain about 13000 frames with the vast majority having over the shoulder shots of about 2 characters.
- **Mosaic Dataset:** I cropped the General dataset with full resolution (720x576) into 5x5 sections and used each one as a separate image. This allowed me to have a dataset with around 1 million images. The motivation for this dataset will later be explained.

To get my data to fit my model, I downsampled every image to a 64x64 resolution.

3.2 Building my Model

I adopted a ground-up approach while building my model. I started with an image size of 28x28x1 (only one channel: black and white) and a relatively simple architecture and capabilities. A brief description of the characteristics of my first Network follows:

- The discriminator consisted of an input image of the size 28x28x1, 3 Convolutional Layers with strided convolutions of stride 2 and one dense layer with 1024 neurons that led to a single output. I used 5x5 Filters.
- The Generator is identical to the discriminator with one major exception: The output and input are swapped, and by doing this, the information flow is inverted, and the down-sampling becomes up-sampling. The input (which was the 1 output in the discriminator) is extended to a 100 inputs that get seeded

with a randomly generated Gaussian noise. To establish a proof of concept of my model, I used the MNIST dataset [17]

I then expanded my model iteratively to 64x64x1 and then 64x64x3 where I used the CIFAR dataset [18]. I purposefully adjusted a few parameters or layers at a time to be able to observe the exact effects of every single change. I took note of changes that caused some major misbehaviour of the Network or proved to yield the most impressive results. These will be discussed in the final chapter.

First, I deliberately didn't refer to the most recent academic consensus on the matter. This served to show the degree of importance to strictly following previous findings, and how flexible models generally are. I then settled on the final architecture detailed in the previous chapter.

3.3 Model Training

When a proof of concept of my model on Cifar10 dataset was achieved, I started running my training algorithm on my General dataset for 20000 iterations. An iteration is here defined as one generator iteration on one batch (meaning 5 critic iterations). Other implementations utilize epochs with each epoch having about 200 batches and running the training for about 100 epochs. I experimented with that variation first but then opted for the use of iterations for simplicity reasons. According to GAN's theory [1], I should set the batch size to be as large as possible to achieve better results. However, limited by device performance, I settled on a smaller size. For my first experiment, I used a batch size of 64 images.

I initially aimed for a higher number of total iterations but remarked after 20000 iterations not only that the Wasserstein loss stagnates but also image quality does not improve significantly, and more overfitting is to be noticed. In figures 3.3 and 3.4, I share my results for this first dataset with the parameters as mentioned earlier, along with the WGAN loss function.

On the basis of the obtained results, I made the following observations that motivated my other experiments:

- **First Observation:** The general dataset generally failed to produce images that depict distinguishable and clear objects that are recognizable and separated from their environment. At the same time, it proved to be capable of capturing the style of an oil painting and produce something relatively novel with it. These results along with the availability of high-resolution images directly extracted from the movie motivated the creation of the aforementioned Mosaic Dataset: Where instead of avoiding the abstraction created by the wide variety of images I further encourage it. The zoomed-in sections allow me to capture higher frequency information without preserving the general spatial structure of the section (I am able to claim this because the general structure was also

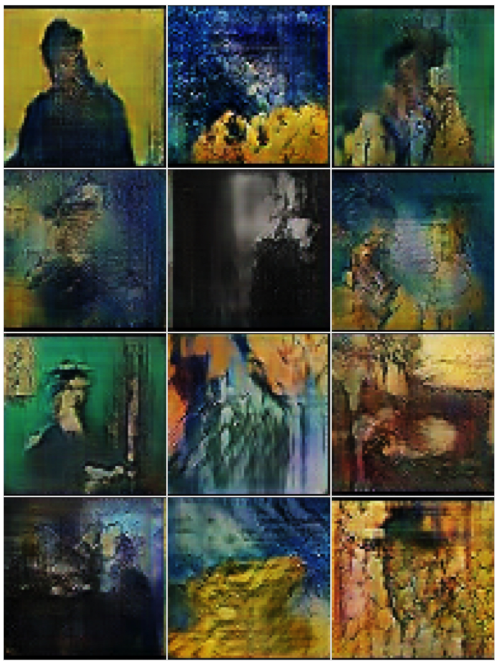
Figure 3.3: 12 Images generated by my GAN using the General dataset. Each image has an original resolution of 64x64 (here scaled for visibility)

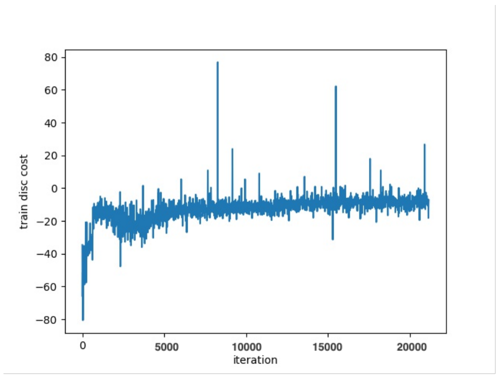

Figure 3.4: The WGAN loss function for the general dataset plotted for 20000 iterations

lost with the General dataset) but in this case deliberately so and they allow me to produce more or less abstract patterns that are much more varied due to the higher number of samples available through sectioning. The choice of the 5x5 partitioning was met as the intermediate point between the maximal amount of sections and a manageable number of images that are trainable in a realistic time frame. Figures 3.5 and 3.6 show the results of running this dataset on my model as well as the loss function.

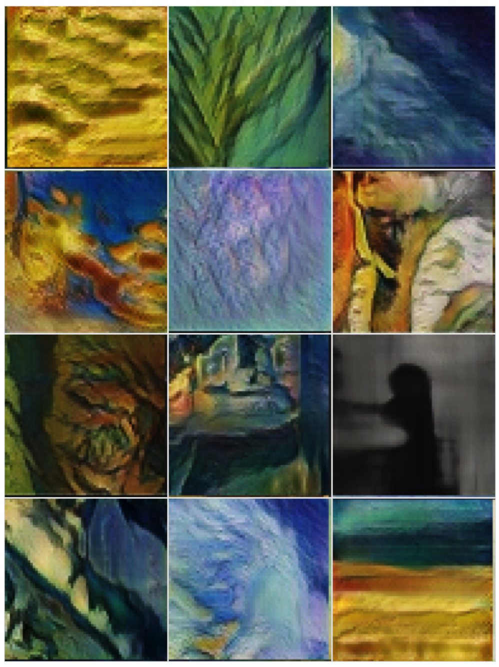

Figure 3.5: Images generated by my GAN using the Mosaic dataset. Each image has an original resolution of 64x64 (here scaled for visibility)

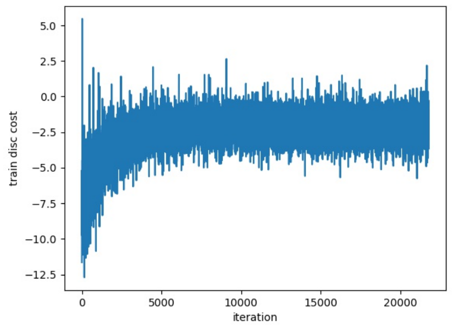

Figure 3.6: The WGAN loss function for the Mosaic dataset plotted for 20000 iterations

- **Second Observation:** The General dataset proved to handle faces and people much better than landscapes which motivated a Portrait dataset.

I then repeated the same experiment, but this time using the Portrait dataset, similarly, figures 3.7 and 3.8 show the results and the loss function.

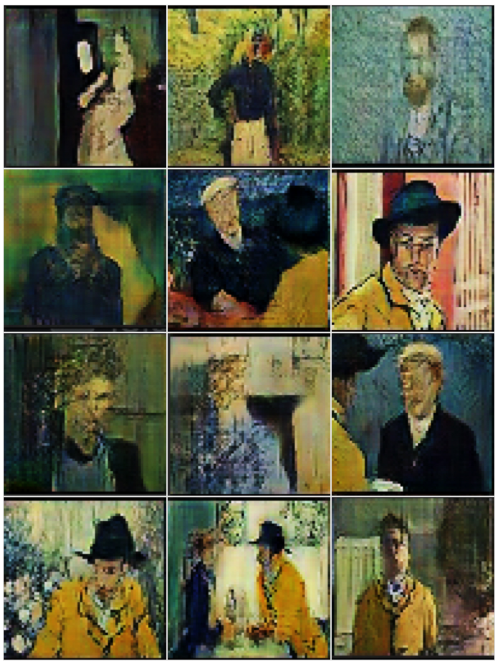

Figure 3.7: Images generated by my GAN using the Portrait dataset. Each image has an original resolution of 64x64 (here scaled for visibility)

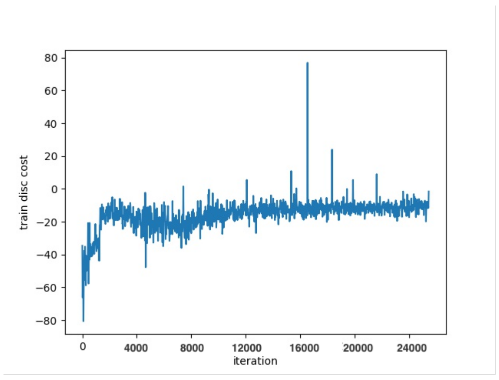

Figure 3.8: The WGAN loss function for the Portrait dataset plotted for 24000 iterations

According to my findings in this experiment, I adjust my model to limit undesired behaviour and encourage better training:

- **Generalization:** When closely cross-examining my generated images and my dataset, I noticed overfitting. To correct for this, I introduce dropout to both Critic and Generator and experiment on different rates to find the values that deliver the best results (in my case 0.5/0.2/0.2 for the Generator and 0.5/0.5/0.2 for the Critic with dropout only between the convolutional layers)

- **The convergence of the loss function:** When examining the Wasserstein loss function, I remarked that the loss doesn't converge properly to 0 but stagnates after about 10000 iterations at a slightly lower value. This may be due to the fact that the critic at this point learns faster than the Generator and is then able to classify images correctly resulting in a non decreasing loss function and thus only provides vanishing gradients that do not enable the Generator to learn appropriately (this, of course, is a mere possible interpretation of the observations made). Following this, I lower down both the critic iterations to 4 from 5 and the learning rate gradually to 0.00005 from 0.0001 after the 10000 iterations mark. These numbers were also obtained through a trial and error approach, and my choice was strictly based on empirical findings. Please note that the mentioned changes were only proven to lead to better performance in my exact setting and my experiments substantiate no proof that these improvements extend to other more general cases.

The results of running my updated model on the Portrait dataset are shared in figure 3.9.

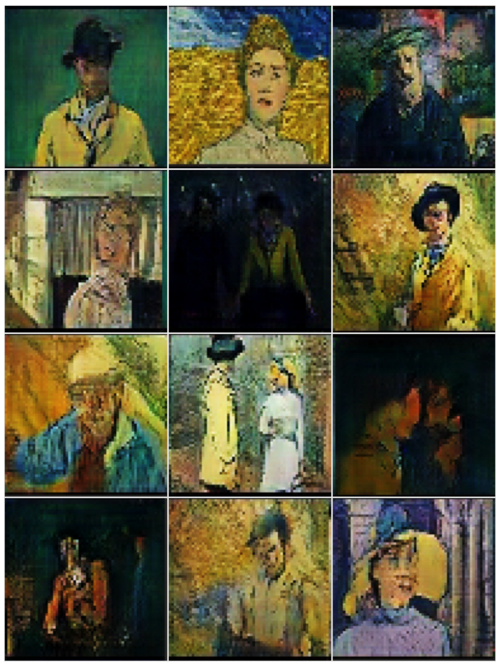

Figure 3.9: Images generated by my improved GAN using the Portrait dataset. Each image has an original resolution of 64x64 (here scaled for visibility)

In my last experiment, I try to test the behaviour of my model on a black and white dataset. I adjust my model accordingly by reducing the input channels and number of parameters; this allows for faster training which in turn allows me to train for more iterations. I settled on a maximum training of 40000 iterations. The best results were still achieved after 20000 iterations, I share these in figure 3.10 as well as the progress of the loss function for the entire 40000 iterations in figure 3.11.

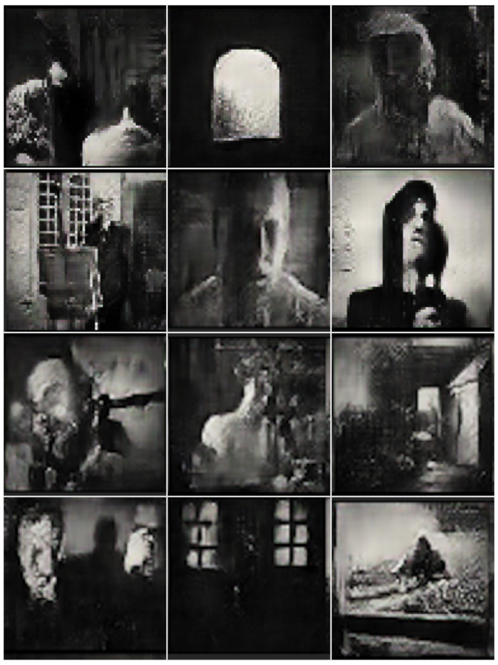

Figure 3.10: Images generated by my GAN using the Black and White dataset. Each image has an original resolution of 64x64 (here scaled for visibility)

One last note to make in this chapter is that I did not encounter mode collapse at any of my experiments with the final model. Mode collapse happens when the Generator generates a single mode from a multi-modal data distribution. This means that we observe limited diversity of generated samples, or even the same sample, regardless of the input noise. Even though the generator might be able to trick the discriminator, it fails to learn to represent the complex real-world data distribution and gets stuck in a small space with extremely low variety. When the generator collapses to this single solution the gradients point in random directions and make it hard to have any further improvement. This happens even if my data is multi-modal which means contains different objects. It is a common problem in GANs. WGANs on the other hand better manage this issue. They provide non vanishing gradients and the critic trains whether the generator is performing or not. This characteristic of the critic is better explained in [4]:

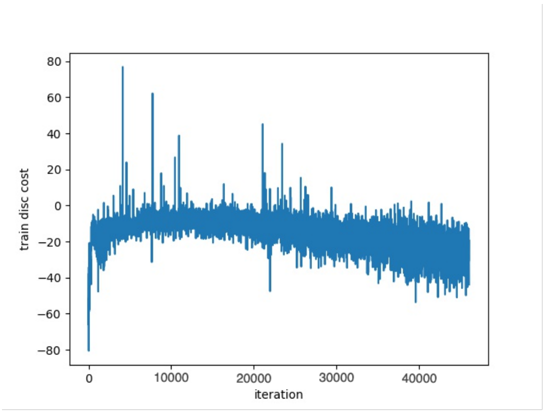

Figure 3.11: The WGAN loss function for the black and white dataset plotted for 40000 iterations

"Perhaps more importantly, the fact that we can train the critic till optimality makes it impossible to collapse modes when we do. This is due to the fact that mode collapse comes from the fact that the optimal generator for a fixed discriminator is a sum of deltas on the points the discriminator assigns the highest values" (p.8).

The interpretation of my different results and a close examination of the behaviour of my model during the experiments follows.

4 Discussion

4.1 Analysis of the generated images

4.1.1 Approach

For the analysis of the results of my different experiments, I concentrated my efforts on two aspects that I deemed descriptive enough for a thorough analysis and that were quickly accessible to me:

- **The loss Function:** (The Wasserstein loss function) Even though the loss function doesn't precisely indicate how well my model is behaving, it gives a general idea of the overall trend of a particular experiment. The Wasserstein loss function is the only function that correlates with sample quality and converges toward a minimum. As for traditional loss functions, they just give an idea about the behaviour of the discriminator which doesn't necessarily correlate with image quality.
- **Image inspection:** Secondly, I heavily rely on the subjective human examination of the sampled images and comparison with the original dataset; for this, I lean on three critical criteria:
  - Variety: My model shouldn't simply memorize my dataset but produce images with great diversity and some degree of originality.
  - Quality: I examine how aesthetically appealing images are and if there are any artefacts or undesired patterns visible on the images.
  - Recognizability: A metric of how recognizable objects and people are on my generated images.

In most GANs and image generation applications, a usual candidate for evaluating generated images is the inception score introduced in [2]. Inception score in a supervised learning setting tries to mathematically describe the concept of realism for a generated set of images by breaking the concept into two criteria:

1. Every realistic image should be recognizable, which means in mathematical terms that the score distribution for it must be, ideally, dominated by one class.
2. At the same time, class distribution over the whole data should be as close to uniform as possible; in other words, a good generator is a generator that generates all different classes of objects.

If both things are true, the score is high. If either or both are false, the score is low.

The inception score, however useful it may be, wasn't applicable in my case due to my data not being labelled and since it is only compatible with datasets that contain objects that the classifier can already detect (common objects in ILSVRC 2014 [19]), which of course was not my case.

The lack of an objective and dataset-independent metric for evaluating generated images motivates further research in this area.

4.1.2 Analysis

Using the subjective image inspection metric, I share my analysis of the output of my model on the different datasets in the following table:

Dataset \Metric	Variety	Quality	Recognizability
General dataset .	Good	Mediocre	Objects slightly recognizable
Portrait dataset .	Mediocre	Very Good	Faces very recognizable
Mosaic dataset .	Very good	No artefacts and smooth images	Nothing is recognizable (Abstract images)
Black and white dataset .	Good	Good	Depends on specific image but mostly recognisable

Please note that my scale to rate the images is nothing but my interpretation of the results after close inspection of hundreds of them, and can easily differ from person to person. Here it serves the function of giving a comparative examination of the different datasets taking into consideration 3 different aspects.

General dataset

The general dataset (the one where I keep almost all frames from the film) provided the worst quality. This may be due to the fact that it contains many images that don't really belong to any class or don't have any other similar images (for example there are shots of random things like lamps and trees, but these don't occur often enough for the GAN to pick up on). These images are to be considered outliers in my data distribution and only lead to throwing my model off and can lead my gradients in directions that impair my training. Simultaneously, this dataset contains black and white images that I forwarded to the network as RGB images with identical values in the three channels. My model could theoretically handle these by considering them a different class of objects, which is more or less what happened as images generated were either entirely in colour or completely black and white. On the other hand, this added another layer of complexity of having another class with its own subclasses, which could have complicated an already difficult task for the critic/generator pair.

Portrait dataset

Examining the generated images from the Portrait Dataset, I am able to derive the following remarks:

During training, the GAN first established a colour scheme for each generated image (first 100 iterations) then a general silhouette of the subjects was established (Iteration 100 to 1000). For the rest of the training, the model builds on this silhouette and adds texture, some degree of details and a distinguishable background. The smallest details are not distinguishable in the end result (such as eyes and eyebrows) which is partly due to the relatively low resolution I was working with (64x64).

Except for a couple of rare cases, the GAN was not able to create new subjects that are not already existing in the original dataset. It was only able to combine elements from different images with a moderate variation. This result pushes me to assume that my model handles subjects as separate classes and not different instances of the same class. My model's ability to generalize is a crucial point that I further address in the second section of this chapter.

Black and white dataset

The black and white dataset contains just like the general dataset various objects, but the output had generally better quality. A Possible explanation for this is that, despite readjustment to the model, the ratio of number of parameters to input size is still higher (with only 1 channel for black and white, which means an input third the size, there needs to be a third as much parameters for the ratio to be equal to the RGB model, which is not the case). Having more parameters to play with could lead to the ability of these parameters to be more descriptive of my data distribution and explain the better image quality (I do not claim that this is indeed the case, I just put forth the assumption for further examination). As adding more parameters increases training time, there is a trade-off. Future work can further look at the effect of the number of parameters on image quality.

Furthermore, I had the opportunity to train the black and white dataset for 40000 iterations due to faster training time. My comparison of the images between 20000 iterations and 40000 iterations is shown in figure 4.1.

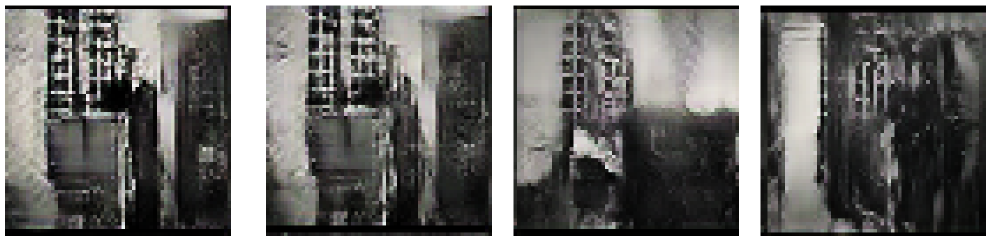

Figure 4.1: The evolution of the behaviour of the generator on one fixed noise sample from iteration 20000 to 40000 (left to right)

The image quality clearly degrades after the 20000 iteration mark, which proves that more training doesn't necessarily correlate with better quality. Considering the higher chance of overfitting with more training, my advice is to regularly inspect generated images and stop the training when any worsening is noticeable. The Wasserstein loss function as shown in figure in 3.11 also shows stagnation after the 20000 iterations and then it slowly starts to diverge from 0 suggesting that the Critic overfits and provides an inaccurate estimate of the data distribution. This is also accompanied by an overall trend of worsening image quality as mentioned in [4] which is also confirmed by my findings.

Mosaic dataset

An important note to make is that with the mosaic dataset, I was able to observe better-behaved cost function values practically converging to 0. My interpretation of this is that this dataset generally had less complexity, and thus, the data distribution was more general, which meant more generated images were deemed real enough for the critic. One important distinction between this dataset and the other ones is the number of samples at hand. Here I have 20 times as much data which proves to be a deciding factor for the overall training. Deep learning has always been proven to work best when big amounts of data are available and GANs are no exception. The more significant number of images in this dataset could also explain this model's ability to generalize better.

4.2 Analysis of my model's behaviour

4.2.1 Robustness

In the context of this bachelor thesis, I had the chance to experiment with a lot of different GAN models and architectures as well as different datasets. I had the chance to tweak most hyperparameters and directly observe the effect on training. With that in mind, I strongly advise any future work to refer to the most recent literature for the choice of significant parameters in the model. These appear to be highly fine-tuned and as is mentioned in [1, 3] a result of long and tedious trials and errors for which many resources are needed. GANs are generally highly unstable and hitting the balance while training two networks at the same time is a complex problem that is yet to be fully understood. So, just a handful of architectures and parametrizations provide excellent and reliable results. The vast majority of my adjustments lead to undesirable outputs, and only a few minor ones were eventually considered. Some parameters have of course more degrees of freedom than others (such as critic iteration and learning rate of the Adam optimizer), but in general, it is best practice to always start with what is recommended in the reliable research papers. My recommendation would be to dedicate more effort into adjusting the dataset at hand to fit the model rather than changing the model to fit the dataset.

4.2.2 Generalization

Secondly, and in a less general sense, overfitting proved to be the bottleneck of the whole process. I implemented dropout, changed the learning rate and reduced the critic iterations halfway through training. These efforts to reduce overfitting, though successful to some degree, did not solve the problem completely. I was observing many generated images that depict exact scenes from the movie just with variations in the positioning of the characters and poses. One might argue that my dataset is just not big enough and diverse enough to allow for a data distribution that doesn't either replicate the data to some degree or produce blurry images. Given enough capacity and too little training data, GANs will tend to overfit. I also noticed that equal representation in my dataset mattered in reducing overfitting: this means that more extended scenes in the film which meant more sample images with very similar information lead to a tendency of the generator to produce images that are similar to these particular scenes. When reducing the number of images from the more extended scenes and making sure data is uniformly distributed between the different settings and characters, I was able to reduce this effect as well as overfitting.

An additional validation subset could better shed light on the degree of overfitting. I did not implement it in my work because of the already not large number of images at my disposal. My initial thought is that I couldn't afford to lose this precious data just for validation. In hindsight, I think that the benefits outweigh the disadvantages and would recommend doing so in future works with GANs.

Newer techniques have since been brought to my attention on how to reduce overfitting; for these, I refer to the zero-centered gradient penalty introduced in [20]. The authors discuss both the used techniques as well as the theoretical background.

4.2.3 Applicability

A significant advantage that GANs have relative to previous deep generative modelling frameworks is the simplicity of sampling images where no Markov chains are needed such is the case for generative Autoencoders. This makes for a relatively non-complex model where I can extract my generator from the GAN and independently sample as many images as needed, which in turn allows for better integration in more significant tasks. Furthermore, my model possesses a very straightforward architecture that trains relatively fast and is very accessible in terms of being able to understand it and implement it for specific needs.

These factors, along with state of the art quality outputs, allow GANs to find many exciting applications and have promising prospects. These can range from image editing [21] to using GANs for security where the discriminator's ability to distinguish between real and fake data can be used to fend off cyber threats as shown in [22] as well as their ability to provide more data and enrich already existing datasets.

My GAN, in particular, can be extended to be able to generate 4-dimensional data (videos) and bring some of the famous Van Gogh paintings to life.

5 Summary

This thesis aimed to implement Generative Adversarial Networks on a new and different dataset and thus better understand these new generative models all while producing images that are similar to human paintings. I first gained insight into GANs thanks to an extensive examination of the wealth of literature, research papers and previous works available. I had the opportunity to experiment on different models and try out different combinations of cost functions, optimizers and learning rates. The choice of adequate architectures and methods came about thanks to the examination of the behaviour of my model and the inspection of outputs. I settled on a deep convolutional neural network [10] for both my generator and discriminator, the Wasserstein loss function [4] as my cost function and an image resolution of 64x64. I also utilized different datasets that I created from the data extracted from the film "Loving Vincent" [9] in order to compare their results and their response to my model.

Based on a qualitative analysis of the generated images I conclude that homogenous datasets perform better with GANs: If the dataset has too many objects with not enough examples for each object, the GAN fails to separate between them and produces blurry images. Additionally, more data generally leads to more varied results and better convergence of the model. I was able to observe this thanks to my mosaic dataset where I had about one million unique images and where I noticed better convergence to 0 for the cost function and fewer traces of overfitting when comparing generated images with my dataset. On the other hand, Wasserstein GANs have proven themselves to be much more stable than the original versions of the GANs. I confidently assert this as I did not encounter one of the most prominent problems of the original GAN, mode Collapse, in my experiments.

I also conclude that GANs are highly sensitive to changes in their architecture and parameters. The smallest of changes that could appear to have little to no effect on behaviour could prove to be crucial for the overall training and contribute to wildly unexpected results. With this, I confirm previous findings relating to GANs [1, 4, 11] and further prove that, for a dataset that is not as well studied as the ones used in these papers, the effect is even more significant. Therefore I strongly advise that recent literature be the starting point when building any GAN. Changes, even when based on logical interpretations and sound reasoning, don't always promise to deliver the expected results. This goes to show further how small our collective understanding of GANs specifically and deep neural networks as a whole currently is and how much more research is needed in this area.

I also address the issue of overfitting and try to find out its causes and reduce its effectiveness. I do acknowledge that my model's ability to generalize is prone to improvement and that future work with purposeful methodology and reference to recent techniques relating to overfitting could provide better and more varied results.

Finally, I was able to produce images that satisfied my initial goals and proved my GAN able to provide impressive results in image generation tasks.

List of Figures

Bibliography

1. Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.

2. Shane Barratt and Rishi Sharma. A note on the inception score. 01 2018.

3. Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5767–5777. Curran Associates, Inc., 2017.

4. Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 214–223, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

5. Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *International Conference on Learning Representations*, 2018.

6. Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *International Conference on Learning Representations*, 2019.

7. Lei Yu, Xiang Long, and Chao Tong. Single image super-resolution based on improved wgan. In *2018 International Conference on Advanced Control, Automation and Artificial Intelligence (ACAAI 2018)*. Atlantis Press, 2018/03.

8. Martín Arjovsky and Léon Bottou. Towards principled methods for training generative adversarial networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

9. Dorota Kobiela and Hugh Welchman. Loving vincent, 2017.

10. Yann LeCun, Patrick Haffner, Léon Bottou, and Yoshua Bengio. Object recognition with gradient-based learning. In *Shape, Contour and Grouping in Computer Vision*, pages 319–, London, UK, UK, 1999. Springer-Verlag.

11. Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. *arXiv e-prints*, page arXiv:1511.06434, Nov 2015.

12. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778, 2016.

13. Forrest N. Iandola, Song Han, Matthew W. Moskewicz, Khalid Ashraf, William J. Dally, and Kurt Keutzer. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and &lt;0.5MB model size. *arXiv e-prints*, page arXiv:1602.07360, Feb 2016.

14. Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas. The earth mover's distance as a metric for image retrieval. *Int. J. Comput. Vision*, 40(2):99–121, November 2000.

15. Masashi Sugiyama, Matthias Krauledat, and Klaus-Robert Müller. Covariate shift adaptation by importance weighted cross validation. *J. Mach. Learn. Res.*, 8:985–1005, December 2007.

16. Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer Normalization. *arXiv e-prints*, page arXiv:1607.06450, Jul 2016.

17. Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.

18. Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research).

19. Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

20. Hoang Thanh-Tung, Truyen Tran, and Svetha Venkatesh. Improving generalization and stability of generative adversarial networks. In *International Conference on Learning Representations*, 2019.

21. Guim Perarnau, Joost van de Weijer, Bogdan Raducanu, and Jose M. Álvarez. Invertible Conditional GANs for image editing. *arXiv e-prints*, page arXiv:1611.06355, Nov 2016.

22. Haichao Shi, Jing Dong, Wei Wang, Yinlong Qian, and Xiaoyu Zhang. SSGAN: Secure Steganography Based on Generative Adversarial Networks. *arXiv e-prints*, page arXiv:1707.01613, Jul 2017.