CycleGAN Comparison on Image-to-Image transformation

Clément Haslé[1], Owein Dourneau[2]

Abstract
Image-to-image translation is a class of vision and graphics problems where the goal is to learn the mapping between an input image and an output image using a training set of aligned image pairs. However, for many tasks, paired training data will not be available. The CycleGAN approach is proposed for learning to translate an image from a source domain X to a target domain Y in the **absence of paired examples**.The aim of this project was to perform a comparison between the initial CycleGAN presented by Jun-Yan Zhu, Taesung Park, Phillip Isola, Alexei A. Efros and an optimized CycleGAN using the U-net architecture on a kaggle challenge named "I'm Something of a Painter Myself". The networks were trained to transform a picture into a Monet's painting. The results show that despite the choices made to optimize the second CycleGAN, it is slightly less performant than the original one on this task.
Source code: https://github.com/Owe1n/ImageToMonet

Video: https://www.youtube.com/watch?v=2MH_hKUsPTQ

Authors
[1] *Computer Science Student at Université de Belfort-Montbéliard, cheha071@ad.liu.se*
[2] *Comuter Science Student at Université de Belfort-Montbéliard, owedo772@ad.liu.se*

Keywords: CycleGAN — ImageToImage — Comparison

Contents

1 Introduction 1
2 Theory 1
2.1 CycleGAN . . . 2
2.2 Objective Functions . . . 2
3 Method 2
3.1 Datasets . . . 2
3.2 Comparison . . . 2
3.3 Improving our CycleGANs . . . 3
3.4 Metrics used to compare . . . 3
3.5 Time optimizing using TPUs. . . . 3
4 Result 4
5 Discussion 4
6 Conclusion 5
References 5

1. Introduction

The subject of generating images thanks to AI is a very hot topic these days and multiple techniques have been developed recently such as diffusion models, transfer style models and GAN's models. GAN is the one that we decided to try and it was convenient thanks to the CycleGAN approach where paired data is unneeded. We compared different approaches of CycleGAN using the FID metric. Trying to vary as many parameters as we could, we were aiming to know in which situation one could be better than another. We firstly used the kaggle's dataset made of 300 Monet's paintings and over 7000 photos. We then have grown the dataset up to 1000 Monet's paintings. We finally ended up trying different sorts of data augmentations to see if it was improving the performance of the models.The idea was to acquire basic knowledge and experience of deep learning for image processing

2. Theory

A GAN (General Adversarial Network) consists of a generator network and a discriminator network. These two can be seen as a forger and a detective, the forger is trying to fool the detective and vice versa. The discriminator is expected to be able to tell if the output has been tampered with, and the generator attempts to make an output that would deceive it.

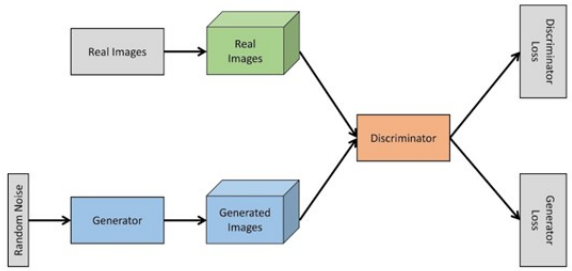

Figure 1. GAN representation

Theoretically, a trained GAN should be able to generate output that appears "real" given enough time. However, it would be probable to run into the "mode collapse" issue which is always producing the same outcome independently of the inputs. This problem can be overcome using CycleGAN model.

2.1 CycleGAN

A CycleGan is composed of 4 Neural Networks :

- 2 Generators (Picture to Monet & Monet to Picture)
- 2 Discriminators (One trying to differentiate the real picture from the generate one and the other one trying to differentiate the real painting from the generate one)

Each generator is trying to fool the corresponding discriminator.

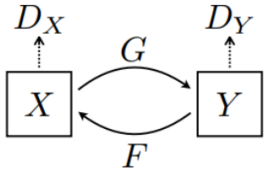

Figure 2. CycleGAN representation

With X the input image, G the generator that transform X into a painting Y. F the generator that transform back Y into a picture. D_Y and D_X are the discriminators.

2.2 Objective Functions

Adversarial Loss

The adversarial losses are applied to both mapping functions. For the mapping function $G : X \rightarrow Y$ and its discriminator D_Y , we express the objective as:

$$\begin{aligned}\mathscr{L}_{GAN}(G,D_Y,X,Y) =& \mathbb{E}_{y\sim p_{data(y)}}[\log D_Y(y)] \\ &+\mathbb{E}_{x\sim p_{data(x)}}[\log 1 - D_Y(G(x))]\end{aligned} \quad (1)$$

where G tries to generate images $G(x)$ that look similar to images from domain Y , while D_Y aims to distinguish between translated samples $G(x)$ and real samples y. G aims to minimize this objective against an adversary D that tries to maximize it, i.e., $min_G\ max_{D_Y}\ \mathscr{L}_{GAN}(G,D_Y,X,Y)$ and $min_F\ max_{D_X}\ \mathscr{L}_{GAN}(F,D_X,Y,X)$.

Cycle Consistency Loss

This metric is the key to overcome the "mode collapse issue".

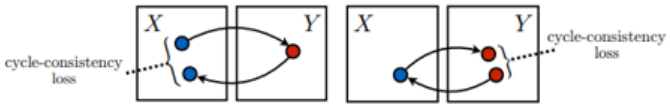

Figure 3. Cycle Consistency Loss representation

The idea of this metric is to compare the 2 times transformed image and painting to the original one. If G is the Monet generator and F the picture generator. Hypothetically, if G and F were perfectly trained, we should always succeed in getting back to the original picture. So training them together implies the objective to not produce the same outputs independently of the inputs since we are supposed to get back to the initial inputs.

3. Method

3.1 Datasets

The initial dataset provided by Kaggle was about 300 monet's paintings and 7038 landscape pictures. Also introducing for us a new optimized format called ".tfrec" which is containing multiple "binaries" images in one file, accelerating the processing in the next steps.

Knowing that only 300 monet's painting won't be enough to have good results, we first scraped the web looking for more monet's paintings. Doing that helped us to grow the dataset up to more than 1000 paintings.

3.2 Comparison

Table 1. Table of characteristics of Original CycleGAN

	Original CycleGan
Generator Architecture	Johnson
Discriminators	PatchGan 70x70
Gradient Descent Optimization	Adam Solver, learning rate 0.0002
Objective Function	Adversarial Loss + Cycle Consistency Loss

Table 2. Table of characteristics of "Home made" CycleGAN

	"Home made" CycleGan
Generator Architecture	U-Net
Discriminators	PatchGan 30 x 30
Gradient Descent Optimization	Adam Solver, learning rate 0.0002
Objective Function	Adversarial Loss + Cycle Consistency Loss + Identity Loss

Here are the two CycleGANs we decided to compare. The first one is an implementation of the CycleGAN described by Jun-Yan Zhu et al. [1].

The second one is the one we handmade using some materials found on the internet and especially on the Kaggle challenge webpage.

Generators architecture

The major difference between those two CycleGAN is the generator architecture. First one uses Johnson et al. [2] This network contains three convolutions, several residual blocks , two fractionally-strided convolutions with stride 1 2, and one convolution that maps features to RGB. Instance normalization is used. Second one uses the U-NET architecture [3] that is supposed to be good for a better generalization of the context.

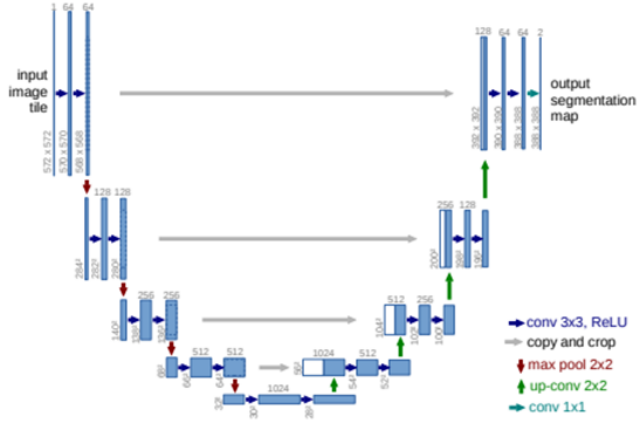

Figure 4. U-NET architecture

Discriminators architecture

Both discriminators use the PatchGan architecture, where the idea is that instead of analyzing the whole image, the discriminators will analyze the image patch by patch and determine for each whether it is fake or not. Here we varied the patch-size.

Objective function

The last difference between these two implementations is the adding of another metric on our CycleGAN. Called Identity Loss, the idea is to handle the case when a painting is given to the painting's generator. Minimizing this function should result in the least possible modifications of the original painting.

3.3 Improving our CycleGANs

By data augmentation

In order to improve our algorithms, we tried to perform some data-augmentations tasks.

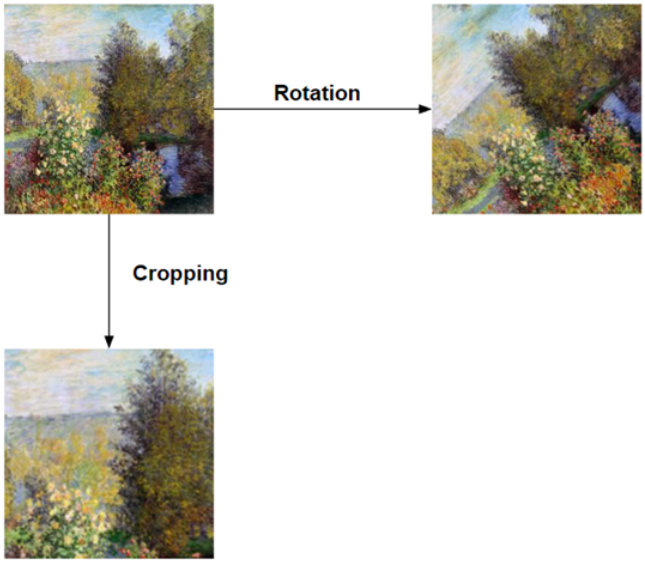

Figure 5. Exemple of data augmentation

By random cropping and rotating , we grew up our dataset to more than 5000 paintings.

By transfer learning

The idea of transfer learning is to fetch an already trained model on a certain task to then fine tune it on our specific task. The Keras API makes the task easier by providing an already trained model and a way to "freeze" the weights and train a model on the top of the first model. And so we tried to use the "Image net" model, initially trained to perform some segmentation tasks on images

3.4 Metrics used to compare

To be ranked on kaggle we had to generate 7000 paintings. To evaluate our submissions Kaggle uses a so-called "Mi-FID" [4] metric which is a specification of the FID. But to keep it simpler to implement, we compared our 2 CycleGAN only with the FID metric. FID, along with Inception Score (IS), are both commonly used in recent publications as the standard for evaluation methods of GANs.

In FID, we use the Inception network to extract features from an intermediate layer. Then we model the data distribution for these features using a multivariate Gaussian distribution with mean μ and covariance $\sum$. The FID between the real images r and generated images g is computed as:

$$FID = ||\mu_r - \mu_g||^2 + Tr(\sum_r + \sum_g - 2(\sum_r \sum_g)^{1/2} \qquad (2)$$

where Tr sums up all the diagonal elements. FID is calculated by computing the Fréchet distance between two Gaussians fitted to feature representations of the Inception network.

3.5 Time optimizing using TPUs.

In order to decrease the time of learning for our Handmade CycleGan, we used the hardware accelerators specialized in

deep learning tasks provided by Kaggle called TPUs (Tensor Processing Units). But it came with restrictions such as only 30 hours weekly utilization and 9 hours at a time in a single session.

4. Result

Model	Parameters	Processing time (hours)	FID
1. Homemade	**200 epochs** | Inputs: **300 paintings** & **300 pictures** | **256*256** resolution	3.5 (on Kaggle)	366.95880126953125
2. Homemade	**10 epochs** | Inputs: **5000 paintings** & **5000 pictures** | **256*256** resolution | Random **crop & rotation + fine tuning**	4 (on Kaggle)	580.7081909179688
3. Homemade	**100 epochs** | Inputs: **1193 paintings** | & **1193 pictures** | **256*256** resolution	3.5 (on Kaggle)	415.88616943359375
1. Original cycleGAN	**200 epochs** | Inputs: **300 paintings** & **300 pictures** | **256*256** resolution	12.5	327.19805908203125
2. Original cycleGAN	**200 epochs** | Inputs: **50 paintings** & **50 pictures** | **256*256** resolution	2	340.86993408203125
3. Original cycleGAN	**200 epochs** | Inputs: **1000 paintings** & **1000 pictures** | Random **64*64 crop** from 256*256	9	295.1295166015625

Figure 6. Table of results

Original CycleGAN

By choosing these two CycleGAN our aim was to try them out and improve the results we will get. To achieve this goal with the original paper-related CycleGAN we tried to be as close as the original settings used to demonstrate the potential of this algorithm in the paper. But being constrained by the time and power of calculation at our disposal, compromises have been made : the main differences are the number of inputs (more than 5000 paintings in the original settings) and their size (512*512 in the original settings).

The first try gave us a decent FID of 327.2 but a processing time of more than 12 hours. To be more efficient, we have then greatly reduced the number of input images in the second try, going down from 300 input paintings to 50. The score of 340.87 obtained was slightly worse than the first score but the processing time was reduced by more than 10 hours. Finally we wanted to greatly increase the number of input images to reduce the FID and so we used 1000 input paintings. The inputs were then randomly cropped to 64*64 so as not to have too much processing time. With these last attempts we obtained our best FID of 295.13 with an acceptable processing time of about 9 hours

"Homemade" CycleGAN

For the first attempt using our own CycleGAN we used similar settings as for the first test of the original paper-related CycleGAN in order to have a direct comparison of the two models. With this first attempt we obtained a FID of 366.96 which is the best score we have with our own CycleGAN model. These parameters are therefore those used to compete with the Kaggle challenge , making us 17th out of 70 participants.

In a second try, using the data augmentation described previously, we managed to recreate a bigger dataset composed of 5000 of Monet's paintings. The counterpart of using such a large dataset was that we couldn't do a lot of epochs, being limited by the process time on Kaggle and the competition that required a process time of less than 5 hours. The number of epochs was only 10 and we also tried to use transfer learning and fine tuning but the final result of this attempt was disappointing, we obtained a FID of 580.71. Given the learning curve, we concluded that a bigger amount of epochs was required.

We returned to more standard parameters for the last attempt by doing a compromise between the number of epochs and the number of input paintings without using data augmentation and fine tuning. We used 1193 input paintings during 100 iterations and obtained a FID of 415.89.

As expected, our own implementation of the CycleGAN is not as effective as the one proposed on github. Every attempt using the original CycleGAN gives better results. On the other hand our model currently takes less time to run than the original CycleGAN but it was run on Kaggle TPUs that we believe are way more efficient than the computer used to run the original CycleGAN.

Figure 7. Generated paintings using different model

On these pictures we observed that the results of the original CycleGAN look more realistic while the ones of our model look more abstract, messier. We also observed artifacts on the Homemade 3 model which is a deformation known to happen while using a U-Net architecture.

5. Discussion

Even if the results were to be expected, we would have liked to improve the results of our own CycleGAN. While experimenting with these models, we encountered several limitations and the first one was the time. Beside the goal of comparing two CycleGAN we also wanted to compete in the Kaggle competition and so we were limited to a process time of 5 hours. For the paper related CycleGAN we didn't respect

this time limitation since the goal was to get the best results but still with our material it was difficult to get good results with less than 10 hours of processing time, even with this well optimized model.

Due to the time limitation, we mainly compared the two CycleGAN regarding their performance. A first improvement of our method of comparison would be to run the two models with the same hardware settings to have a real comparison of the ratio between time and efficiency of these two models. It would also be interesting not to limit our model to 5 hours of process time.

By not limiting the processing time we believe that we would be able to increase the performance of our own CycleGAN by increasing the number of epochs. To improve our results, we could also try to run our model with the settings that worked best for the original CycleGAN. In general, it might have been a good idea to set a training plan to execute for both models. The comparison would have been more direct, and we could have concluded on how the differences affect the results depending on the settings and inputs.

Another limitation comes from our dataset of real images. It was mainly composed of landscape pictures but also contained photos of people and animals or close shots. Even if money painted some close shots, he mainly painted landscapes and thus the results obtained with this type of images are not satisfying. Our dataset containing these images is then both a negative point for the training and the evaluation. Another way of improvement would be to sort the dataset to get rid of these types of images before training and evaluating. With the example below we can also clearly see the artifacts produced by our CycleGAN with the U-Net architecture.

Figure 8. Badly handled image

Finally, we would like to discuss the relevance of our evaluation methodology for this project. Indeed, the FID is a purely mathematical method that gives an indication on the similarity of the generated and real images but in the field of art we feel like evaluating a painting is very subjective. In addition the painter's style changes over the years and lots of Monet's paintings look very abstract and others look very realistic, see below an example of two paintings from our dataset. This way the best result of our own CycleGAN, represent beforehand "HomeMade 1", may seem more accurate if we compare it to the more abstract paintings but the best one of the original CycleGAN, "Github 3", would be more accurate if we compare it to more realistic paintings. To face this problem, we could stick to a particular style of the painter or a certain period.

Figure 9. 2 different monet styles

6. Conclusion

To conclude this project, we would say that it has shown the general efficienty of the CycleGAN approach to perform Image to Image transformation tasks. Even if the "homemade" implementation did not overcome the original version, it has shown some interesting results and also some limits such as the artifacts apparition with the U-net architectures. The project also showed that augmenting the size of the dataset was a good way to improve the result of the CycleGAN. Finally, creating our own CycleGAN with Keras and using it via a web application was an effective way to understand the basics of deep neural networks and how to use them in real applications.

References

[1] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *arXiv.org*, Aug 2020.

[2] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. [pdf] perceptual losses for real-time style transfer and super-resolution: Semantic scholar. *undefined*, Jan 1970.

[3] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *arXiv.org*, May 2015.

[4] I'm something of a painter myself. *Kaggle*.