Cycle Generative Adversarial Nets for Style Transfer of Monet Paintings

Solmaz Mohammadi
solmaz.mohammadi@studenti.unipd.it

Nicla Faccioli
nicla.faccioli@studenti.unipd.it

Abstract

This article aims at learning the style of the impressionist painter, Claude Monet. Since the generative adversarial model has its limitations when it comes to more complex computer vision problems, we will be using a variation of generative adversarial models, cycle consistent GAN to capture the "impression" of an artist from their surroundings. CycleGAN is an unpaired image to image translation using cycle consistent adversarial nets.

1. Introduction

The focus of this paper is to reproduce the style of Claude Monet, the influential impressionist painter in the nineteenth century. Claude Monet's style is unique and recognizable by his short and thick strokes that capture the essence of the subject. Monet's work is particularly interesting because of the strong logarithmic correlations within his paintings that makes it possible to generate new Monet-like images. [9] This project attempts to transfer the painter's style into existing real-life photos. Style transfer is a computer vision technique that aims to combine two images together. In particular, given a content image (i.e. is interesting for its content) and a style image (i.e. is interesting for its style), the task is to blend them together in order to obtain as output a third image which will represent the content of the first image in the style of the second one. Therefore, we aim to learn a mapping function G: X⟶Y such that the distribution of G(X) is inseparable from the disribution of Y using the adversarial loss (i.e. the discriminator). To further constraint the generator, we pair it with an inverse map F:Y⟶X and use cycle consistency loss to enforce F(G(X))∼X. [19]

2. Related Work

In **Generative Adversarial Nets (GANs)**, two models are trained simultaneously: a generative model G the captures the data distribution, and a discriminative model D that estimates the probability that a sample is from the original data rather than generated by G [6]. The training procedure for G is to maximize the probability of D making a mistake. Therefore, it corresponds to a 2-player minimax problem in which generator tries to "trick" the discriminator. This framework is then the core idea of many generative model applications including cycle consistent GANs. However, the simple GAN is under-constrained for style transfer problem [19]. We apply a cycle consistency loss objective to further condition GAN.

Neural Style Transfer (NST) First formulated and demonstrated by *Gatis et al.* [5], NST involves the use of a CNN (e.g., VGG-19) pretrained in object recognition, an image to be transformed (p) and an image from which the style has to be copied (a). The image p is fed through the CNN and the resulting output P is the "content representation" of p. Then a is also fed to the CNN and the sum of the outputs of every layer of the weighted loss function becomes the "style representation" A. A key difference between P and A is that for the former only the output of the last layer is relevant while, for the latter, the output of every layer is considered. In A usually the outputs of the first layers are the ones with the higher weight. The full objective of NST is then, starting from p with some white noise added, to generate an image x that minimizes the differences between its content (F) and p's content (P) and also between its style (G) and a's style (A). Hence, the general loss function to be minimized has the following structure:

$$\mathcal{L} = \alpha|F - P| + \beta|G - A|$$

where $\frac{\alpha}{\beta}$ is usually between 1×10^{-3} and 1×10^{-4}.

Paired Image-to-Image Translation: In some applications the aim is not to generate the image from scratch but rather, "translate" an image from a domain X, to another domain Y. This can be done by conditioning GAN to learn a dataset of paired images and the function "translating" them. This approach introduces cGAN, where the given the input, the output is conditoned by a set of constraints. The conditional GANs learn a mapping from observed data x and random noise vector z, to the output y, G:x,z→y [8]. cGANs in the context of image translation can be generalized to cover various problems. Image prediction from a map [17], future frame prediction [11], and image generation from sparse annotations are some of the examples [13]. Image-to-Image mappings are also the core idea for style

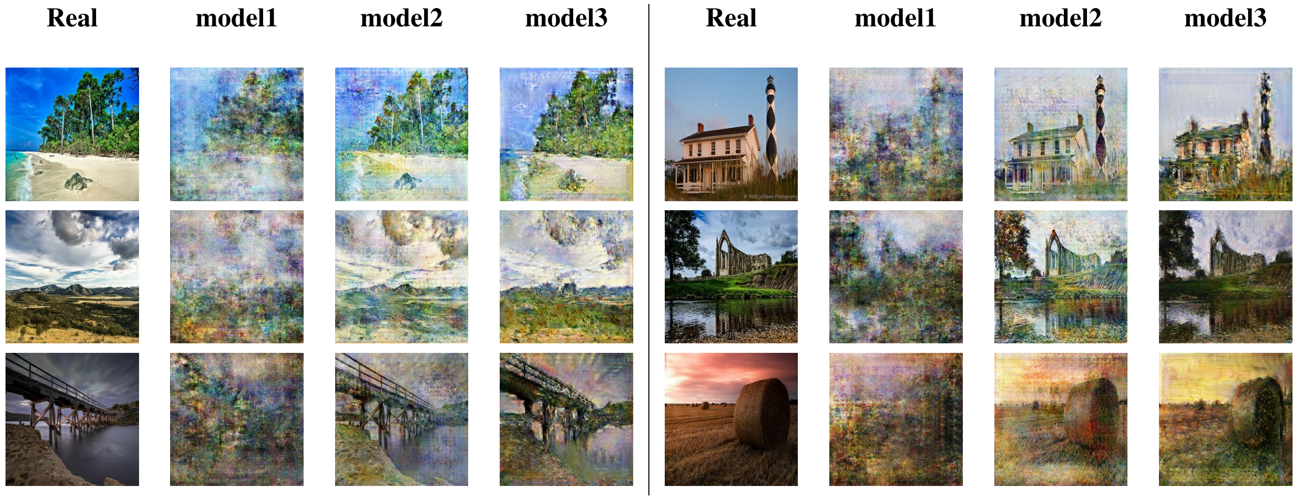

Table 1. **Results**. Comparison between the generated images from different models.

transfer problem, but with GAN applied unconditionally [10].

3. Dataset

The dataset was collected from the Kaggle competition [1], a platform in which challenges are proposed and where we got the idea for this project. The dataset is composed by:

- A set of 300 Monet paintings sized 256x256 in both JPEG and TFRecord format;
- A set of 7028 photos sized 256x256 in both JPEG and TFRecord format.

We used the Monet paintings to train our model so that it could learn Monet style and imitate it. Then we applied the learned style over the set of real photos in order to generate new imges that could resemble real paintings. The size of all the images was already the right one for the computation, hence no resizing was needed.

4. Method

Given the real-life images $\{x_i\}_{i=1}^{N}$ where $x_i \in X$ and monet paintings $\{y_j\}_{j=1}^{M}$ where $y_j \in Y$, donate the data distributions as $x \sim p_{data}(x)$ and $y \sim p_{data}(y)$. We aim to learn the mappings between the two domains X and Y using two generative models $G : X \longrightarrow Y$ and $F : Y \longrightarrow X$. Additionally, we also include two adversarial discriminators D_X and D_Y where task of D_X is to discriminate between {x} and translated images {F(y)}. Similarly, D_Y aims to distinguish between {y} and {G(x)} [19]. The objective of our work is to minimize two losses: *adversarial loss* [6] which helps to close the gap between the data distribution of target domain and the generated images; and *cycle consistency loss* to keep the results of the two mapping functions consistent.

4.1. Adversarial Loss

To learn the distribution of $x \sim p_{data}(x)$, we first define a prior on input noise variables $z \sim p_z(z)$ and define a mapping $G(z, \theta)$ where θ is the set of parameters. The generative model G is often trained using a convolutional neural network [12]. We also define a discriminator D (also trained using a CNN) that instead of a single scalar [6], outputs an image with decreased dimentionality where pixels that are more likely to be in data distribution $p_{data}(x)$ have higher values. We train both models simultaneously in a two-player minimax approach that can be expressed as:

$$\begin{aligned}\min_G \max_D V(D, G) = &\mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] + \\ &\mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))]\end{aligned} \tag{1}$$

4.2. Cycle Consistency Loss

In theory, adversarial training can fully adapt the target domain distibution and give prominant results. However, real-world datasets can be quite noisy, and in our case the cardinality of the two datasets were considerably uneven. Therefore, the learning process can fail as a network can map the same set of input images to any random permutation of images in the target domain, where any of the learned mappings can induce an output distribution that matches the target distribution [19]. For this reason, adversarial losses alone cannot guarantee that the desired output is given by the learned function. To further constraint the mapping function, we introduce the *cycle consistency loss*: For each sample $x \in X$, the following cycle $x \rightarrow G(x) \rightarrow F(G(x)) \approx x$ should bring x to the original image. Similarly, for each $y_i \in X$, the consistency of $y \rightarrow F(y) \rightarrow G(F(y)) \approx y$ is also satisfied [19]. The

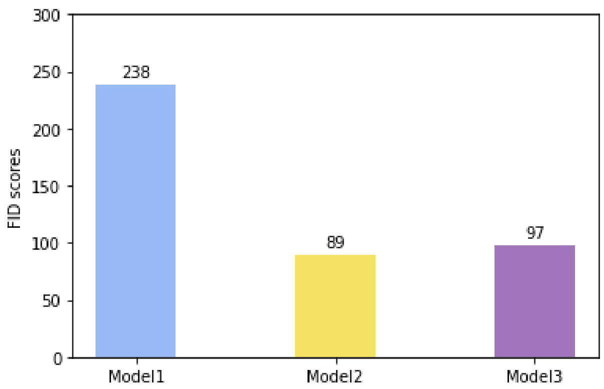

Figure 1. **FID**. Results of FID evaluated over the different models that we experimented.

objective is then defined as:

$$\begin{aligned}\mathcal{L}_{cyc}(G,F) =&\mathbb{E}_{x\sim p_{data}(x)}[\|F(G(x))-x\|_1]+\\ &\mathbb{E}_{y\sim p_{data}(y)}[\|G(F(y))-y\|_1]\end{aligned} \tag{2}$$

4.3. Implementation

Network Architecture We used a *U-net CNN* for our generative network. A *U-net* is a fully connected convolutional neural network that was first introduced for biomedical image segmentation [15]. The standard procedure for our experiments is to first downsample the data (downsampling data is the process of reducing the width and height of the image by strided convolutions), followed by applying several convolutional layers to obtain a deeper image transformation. To generate the final results, we upsample (conversely, increasing the height and width to reach the desired output size) using two convlutional layers with $\frac{1}{2}$ strides. We also applied instance normalization [16] to all layers and established the skip connections in the *Unet*. For the discriminator we used a PatchGAN model [4]. APatchGAN is a type of discriminator for GANs which only penalizes structure at the scale of local image patches. Our PatchGAN discriminator tries to classify if each 30 × 30 patch in a given image is real or fake.

5. Experiments

The core of all of our experiments follow the explained objectives and have a fully-connected *U-net* structure. We focus on building different implementations of the *U-net* model and compare the results. Out of all the experiments, we selected three main models: The first model (i.e. `model1`) uses a naive approach in generative models. It has 5 downsampling and 5 upsampling layers with filter size 8 × 8 and no intermediate layers. In downsampling, we used *maxpooling* with 2-stride and the activation *leakyReLU*. This model has a high level of noise and the results obtained were not satisfactory. The second model () was inspired by the Kaggle competition suggested framework [1]. It has a smaller filter size (4 × 4) and more layer of both downsampling and upsampling such that it reaches the pixel-sized dimensionality. It uses dropout in three of the upsampling layers and the activation functionn is *leakyReLU*. This model has the benefit of having more precision as a generator and in some cases generates prominent results. For the third model, (`model3`) we removed the dropout and instead of reducing the size to 1 pixel in downsampling, we stop at size 64 × 64 and add several *transformer* blocks without size alternation. Although this model has a noticeable degree of noise in the generated images, in some examples, it is more "monet-esque" to the eyes of the viewers and it contains the brushing style of the painter considerably more than `model2`.

5.1. Evaluation metrics

In order to evaluate the results of our work we looked for some metrics that could help us in this task [2]. In order to apply them we used already existing code on GitHub [18].

Fréchet Inception Distance (FID) Introduced by *Heusel et al.* [7], FID measures the distance between the feature vector of the generated paintings and the feature vector of the real paintings. Therefore, a lower FID score represents a better performance of the generator since the images it creates are similar to the real ones. The formula to calculate the FID score is the following:

$$\begin{aligned}FID(r,g) =&\|\mu_r-\mu_g\|_2^2+\\ &Tr(\textstyle\sum_r+\sum_g-2(\sum_r\sum_g)^{\frac{1}{2}})\end{aligned} \tag{3}$$

where $(\mu_r, \sum_r)$ is the multivariate normal distribution estimated from a specific layer of Inception Net features for real images and $(\mu_g, \sum_g)$ is the multivariate normal distribution estimated from a specific layer of Inception Net features for generated images.

Number of statistically-Different Bins (NDB) Introduced by *Richardson and Weiss* [14], this metric is applied to the image pixels and, unlike FID, does not rely on a representation of those images. This evaluation method is based on the observation that given two set of samples originated from the same distribution, the number of samples that falls into a given bin should be the same up to sampling noise. More formally, if $I_B(s)$ is an indicator function for bin *B*, i.e. $I_B(s) = 1$ if the sample s falls into bin B and $I_B(s) = 0$ otherwise, and given s_i^p and s_j^q respectively N_p samples from distribution *p* and N_q samples from distribution q, then if $p = q$ then we expect to be $\frac{1}{N_p}\sum_i I_B(s_i^p) \approx \frac{1}{N_q}\sum_j I_B(s_j^q)$. When this is not true for a given bin, they are said to be *statistically different*. The test is performed on all bins and then the number of statistically-different bins divided by the number of bins is the final result. We used this metric in order to measure

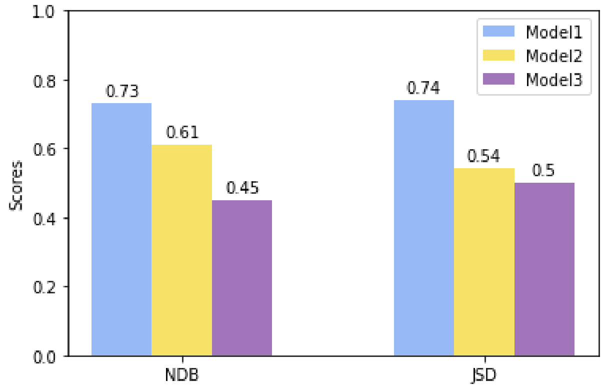

Figure 2. **NDB and JSD**. Results of NDB and JSD evaluated over the different models that we experimented.

the diversity of the generated images and to understand if the model was encountering mode collapse by generating similar images.

Jensen-Shannon Divergence (JSD) As reported in [3], JSD is a smoothed and simmetrized version of the Kullback divergence amd measures the divergence between two probability distributions p and q. It is defined as:

$$D_{JS}(p\|q) = \frac{1}{2}D_{KL}(p\|\frac{p+q}{2}) + \frac{1}{2}D_{KL}(q\|\frac{p+q}{2})) \quad (4)$$

This can be used in order to understand the diversity of the generated in a slightly different way with respect to NDB.

6. Conclusions

We applied the three models we implemented to the original photos and, as it's possible to see in Table [1], the results differ a lot from one model to the other. From a visual point of view, the first model we proposed generates output with a huge degree of noise. Hence the result is a set of images in which is impossible to distinguish any element of the original photos. `Model2` on the other hand is able to generate images that better represent the content of the original photos and also applies some of Monet style. With respect to `model1`, with `model2` it is possible to appreciate a relevant improvement. Finally, the third model enhance a bit more the painter style since it's possible to appreciate some hint of strokes and brushing. The issue with `model3` is the high degree of noise but overall seems to create images that better imitate the orginal paintings. Also the measurment performed by the metrics we applied seems to confirm what we were able to visually appreciate: the quality of the image results to be better with `model2` since it presents a lower level of noise, but from a diversity point of view, `model3` performed in a better way.

References

[1] Kaggle competition: I'm something of a painter myself. `https://www.kaggle.com/competitions/gan-getting-started`.

[2] Ali Borji. Pros and cons of gan evaluation measures, 2018.

[3] Jop Briët and Peter Harremoës. Properties of classical and quantum jensen-shannon divergence, may 2009.

[4] Ugur Demir and Gozde Unal. Patch-based image inpainting with generative adversarial networks, 2018.

[5] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. A neural algorithm of artistic style, 2015.

[6] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[7] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium, 2017.

[8] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks, 2016.

[9] Jaron Kent-Dobias. Log-correlated color in monet's paintings, 2022.

[10] Chuan Li and Michael Wand. Precomputed real-time texture synthesis with markovian generative adversarial networks, 2016.

[11] Michael Mathieu, Camille Couprie, and Yann LeCun. Deep multi-scale video prediction beyond mean square error, 2015.

[12] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks, 2015.

[13] Scott Reed, Zeynep Akata, Santosh Mohan, Samuel Tenka, Bernt Schiele, and Honglak Lee. Learning what and where to draw, 2016.

[14] Eitan Richardson and Yair Weiss. On gans and gmms, 2018.

[15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[16] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization, 2016.

[17] Xiaolong Wang and Abhinav Gupta. Generative image modeling using style and structure adversarial networks, 2016.

[18] Yahui Liu Yang Li. Gan-metrics. `https://github.com/yangco-le/GAN-Metrics`, 2020.

[19] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks, 2017.