Neural Algorithm and Cycle GAN : Artistic Style Comparison

Ayoub GHRISS
ayoub.ghriss@ens-paris-saclay.fr
Object Recognition and Computer Vision
Master MVA

Introduction

Our goal is to test the performance of the Cycle GAN for artistic style transfer. The reference model chosen is the Neural Algorithm by Gatys (1). It first appeared in 2015 and has since knew popular success that lead to the development of many applications for smart-phone (Prisma) and the website DeepArt, which perform the computations on-line.
In this report, we will provide an overview of each of these models' methods before providing the implementation details and our results. In particular, we present a cross-comparison where we tested each method's results using the other method's objective.

1 Methods

1.1 Gatys' Neural Algorithm

Given a content image C and style image S, the main idea of this method is to find an image that matches the content features of C and the style features of S. The features spaces used are provided by convolutional and pooling layers of a pre-trained Convolutional Neural Network for object recognition (VGG-Network in our case).

An input image $\vec{x}$ is encoded in each layer of the CNN by the filter responses to that image. A layer l with N_l distinct filters has N_l feature maps each of size M_l. Hence, the responses in a layer l can be stored in a matrix $F^l \in \mathcal{R}^{N_l \times M_l}$ where F^l_{ij} is the activation of the i^{th} filter at position j in layer l.

Content Loss Let $\vec{p}$ and $\vec{x}$ be the original content image and the image that is generated and P^l and F^l their respective feature representation in layer l. We define the squared-error loss between the two feature representations

$$\mathcal{L}_{content}(\vec{p}, \vec{x}, l) = \frac{1}{2} \sum_{i,j} \left(F^l_{ij} - P^l_{ij} \right)^2 \ . \tag{1}$$

Style Loss The model builds a style representation that computes the correlations between the different filter responses, where the expectation is taken over the spatial extend of the input image. These feature correlations are given by the Gram matrix $G^l \in \mathcal{R}^{N_l \times N_l}$ for layer l:

$$G^l_{ij} = \sum_k F^l_{ik} F^l_{jk}. \tag{2}$$

So let $\vec{a}$ and $\vec{x}$ be the original style image S and the image that is generated and A^l and G^l their respective style representations in layer l. The contribution of that layer to the total loss is then

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} \left(G^l_{ij} - A^l_{ij} \right)^2 \tag{3}$$

and the total loss is

$$\mathcal{L}_{style}(\vec{a}, \vec{x}) = \sum_{l=0}^{L} E_l \tag{4}$$

1.1.1 Implementation*

To generate the images that mix the content of a photograph with the style of a painting we jointly minimise the distance of a white noise image from the content representation of the photograph in one layer of the network and the style representation of the painting in a number of layers of the CNN.

To generate a texture that matches the style of a image S, we use gradient descent from a white noise image to find another image that matches the style representation of the original image. So let $\vec{p}$ be the photograph and $\vec{a}$ be the artwork. The loss function we minimise is

$$\mathcal{L}_{total}(\vec{p}, \vec{a}, \vec{x}) = \alpha\mathcal{L}_{content}(\vec{p}, \vec{x}) + \beta\mathcal{L}_{style}(\vec{a}, \vec{x}) \tag{5}$$

We use $\alpha = 1$ and $\beta = 500$. While the authors pointed out that best results can be achieved if the average pooling was used to retrieve the VGG features, in our experiment for Van Gogh style, the max pooling gave better results. For the minimization operations we let the Adam optimizer of Tensorflow Python library handle the gradient derivations.

Our base implementation started from a functional version* to borrow the VGG exploitation part. It was later heavily modified to an objective one and rendered more efficient to visualize losses and results.

1.2 Cycle Generative Adversarial Networks

Given two sets A and B, the Cycle GAN core goal is to learn two mappings $G : A \to B$ and and $F : B \to A$ given training samples : $\{a_i\}_{i=1}^{N} \in A$ and $\{b_j\}_{j=1}^{M} \in B$. We denote $a \sim p_A$ and $b \sim p_B$ the data distributions. The model introduces two adversarial discriminators D_A and D_B, where D_A aims to distinguish between images $\{a\}$ and translated images $\{F(b)\}$; the same analogy applies to D_B. The objective contains two types of terms: *adversarial losses* for matching the generation distribution to the target data distribution; and *cycle consistency losses* for preventing the mappings G and F from contradicting each other.

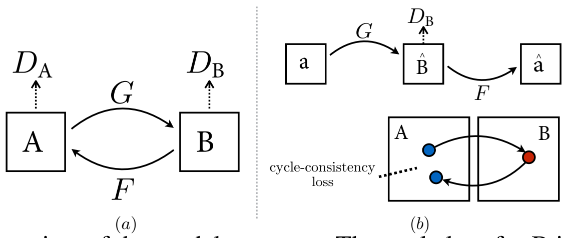

Figure 1: An overview of the model structure. The cycle loss for B is analogous to A

Adversarial Loss The adversarial loss applies to both mapping functions. For the mapping function $G : A \to B$ and its discriminator D_B:

$$\mathcal{L}_{\text{GAN}}(G, D_B, A, B) = \mathbb{E}_{b\sim p_B}[\log D_B(b)] + \mathbb{E}_{a\sim p_A}[\log(1 - D_B(G(a))], \tag{6}$$

where G tries to generate images $G(a)$ that look similar to images from domain B, while D_B aims to distinguish between translated samples $G(a)$ and real samples b. G aims to minimize this objective against the adversary D_B that tries to maximize it. Hence the name 'adversarial' that reflects the similarity with models in game theory. We introduce a similar adversarial loss for the mapping function $F : B \to A$ and its discriminator D_A as well: i.e. $\min_F \max_{D_A} \mathcal{L}_{\text{GAN}}(F, D_X, Y, X)$.

Cycle Consistency Loss For each image a from domain A, the image translation cycle should be able to bring a back to the original image: $a \to G(a) \to F(G(a)) \approx a$. We call this *forward cycle consistency*. Similarly, we define the backward cycle consistency : $b \to F(b) \to G(F(b)) \approx b$. We can incentivize this behavior using a *cycle consistency loss*:

$$\mathcal{L}_{\text{cyc}}(G, F) = \mathbb{E}_{a\sim p_A}[||F(G(a)) - a||_1] + \mathbb{E}_{b\sim p_B}[||G(F(b)) - b||_1] \tag{7}$$

1.2.1 Implementation*

The objective is simply the sum of the two GAN losses and the Cycle Consistency loss. We implement the same architecture used in (2). Since we used training sets of 256x256 size images, the mapping are modeled with the 9 residual blocks structure, with instance normalization. For the discriminators, we used the Markovian Discriminator PatchGANS (3) which classifies whether 70x70 overlapping image patches are real or fake. Our implementation was mostly based on those provided in the paper's authors page. The implementation part was to tweak different parameters and understand the structure in order to build our own discriminator that was used for the cross-comparison later.

2 Results

In our experiment, we test the performance on the Van Gogh style, we use the same database used by the referenced article, with 400 painting.

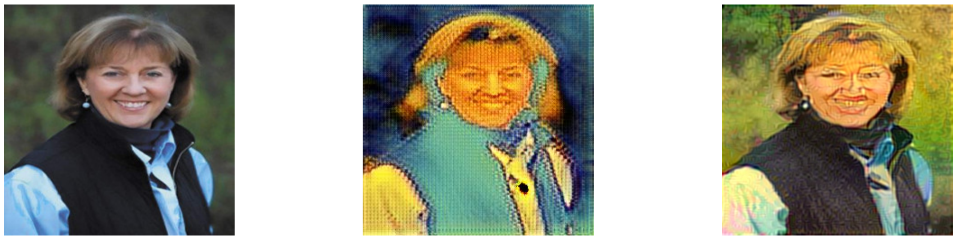

Figure 2: An example comparison from the output of both models

2.1 Neural Judge

Figures 3 and 4 in the Appendix plots the content loss and style loss for the images generated by GAN vs Generated by the Neural Algorithm. We used 100 iterations for CycleGAN and 200 for Neural Method Despite the nearly identical starting points, the GAN performs poorly when it comes to the $\mathcal{L}_{content}$ and $\mathcal{L}_{style}$ losses. In the implementation, we added a total variance loss to the Neural loss in order to smooth the output. It seems that the GAN only wins against the Neural method in the smoothness of the images.

2.2 GAN Judge

In the second step, we implemented a miniature version of GAN where the purpose was the train the discriminators D_A and D_B, with A the set of Van Gogh Paintings. For the set B that we discriminated against, we used 400 real photography pictures mainly for portraits and landscapes. Afterwords, we used the trained discriminator to calculate the average GAN loss for the set generated by the GAN Versus the set generated by the Neural Method. For a fake image x that is supposed to be similar to images of A, we define this loss as:

$$\mathcal{L}_{\text{GAN}}(x) = [D_B(x)]^2 + [1 - D_A(x)]^2 \tag{8}$$

Since we trained D_Q (for $Q \in (A, B)$) to be near 1 on Q and 0 outside Q, $\mathcal{L}_{\text{GAN}}$ should be higher for x different than elements of A.

We used 100 GAN and 100 Neural images for the comparison. the table 1 below provides the results. It shows that even using GAN's own criteria, the Neural method's results are almost indistinguishable from original paintings using GAN discrimination. The Neural set was generated using styles from the training set, which explains why it performs better than the test set of Van Gogh paintings.

Table 1: GAN Discriminator Results

Set	Size	Average GAN Loss
Test Van Gogh set	200	0.43
GAN Generated	100	0.45
Neural Generated	100	0.39

3 Conclusion

From the different tests performed, and as it has been explicitly showed during the presentation, when it comes to the artistic style transfer, the Neural Method in (1) has the upper hand. The main drawback of the CycleGAN was its inability to transfer shapes or brush strokes of paintings.

However, while this comparison seems interesting, it remains a relatively unfair one. The CycleGAN, and the GAN models in general, apply to a wide range of problems including mappings that involve texts and audio signals. Using this multipurpose aspect of the GANs, an ambitious attempt that we have started was to apply the CycleGAN on the features spaces of the VGG network. The goal was to generate fake contents and styles representations and then trace back the solution image. The most important constraint was mainly a computational one. Which points out another advantage og the Neural method, which was its low resources requirements.

Appendix

Comparison plots

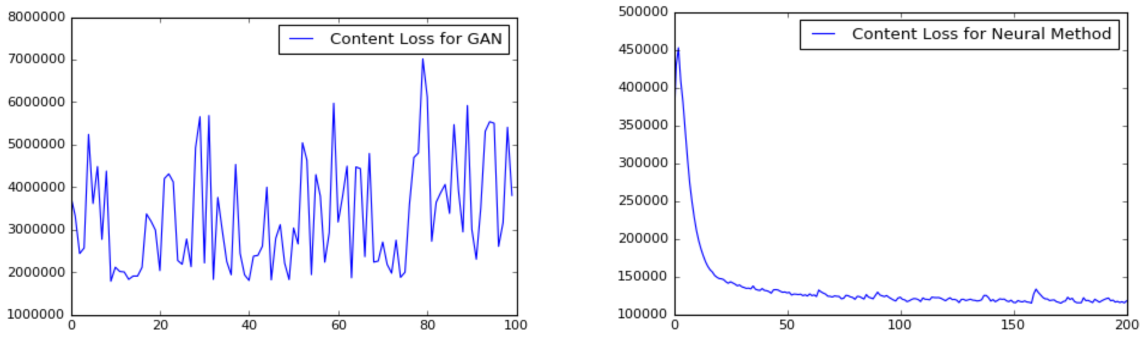

Figure 3: An overview of the model structure. The cycle loss for B is analogous to A

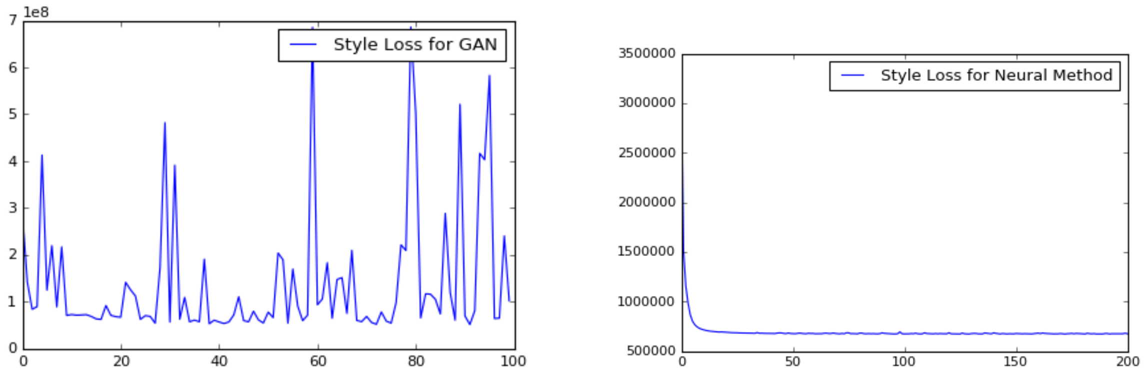

Figure 4: An overview of the model structure. The cycle loss for B is analogous to A

References

[1] Leon A. Gatys & Alexander S. Ecker & Matthias (2015) A Neural Algorithm of Artistic Style *arXiv:1508.06576*.

[2] Jun-Yan Zhu & Taesung Park & Phillip Isola &Alexei A. Efros (2017) Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks *arXiv:1703.10593*.

[3] Phillip Isola & Jun-Yan Zhu & Tinghui Zhou & Alexei A. Efros (2016) Image-to-Image Translation with Conditional Adversarial Networks *arXiv:1611.07004*.

[4] Antonia Creswell & Tom White & Vincent Dumoulin & Kai Arulkumaran & Biswa Sengupta & Anil A Bharath(2016) Generative Adversarial Networks: An Overview *arXiv:1703.10593*.