Style Transfer with GANs

Xavier de Juan Pulido
Universitat de Barcelona

Lorenzo Andrés Vigo del Rosso
Universitat de Barcelona

Abstract

Style transfer is a technique that preserves the content of an input image but changes its representation by adapting the content to a style characterized by a predefined set of samples. In this paper, we will study two methods that implement this technique, which use architectures based on Convolutional Neural Networks (CNN) and Generative Adversarial Networks (GAN). Finally, we will do a quick qualitative comparison of the results produced by both approaches. This work aspires to be a simple introduction to the more complex issue of comparing the quality of auto-generated samples. Future work will focus on the ultimate goal of implementing and defining heuristics or validation techniques that numerically show the presumed improvement in quality in those samples generated by GANs respect to the one attained by CNN samples.

1 Introduction

Whether Machine Learning models will ever be capable of overtaking the role of artists in our society or not is a recurrent question that still has no definite answer. It is true that nowadays' models continue to fall short on creativity. However, there are several models that are able to produce great results by *learning* and imitating art.

We could enumerate a couple notorious mentions such as GPT3-based models writing essays and short stories and models specialized in generating reggaeton songs. In this work, we will focus on models which area of expertise lies on a canvas.

The models that will be seen throughout this paper are able to separate the content and the style of a picture. These models learn different styles from the pictures that are handed to it as training set. Afterwards, they can extract the content from an input image and generate a new sample that expresses said content in a pre-obtained style. This is known as *style transferring*.

The effect of style transfer can lead to various and curious results. For example, we can turn a photograph into a painting and vice versa or *translate* a painting of an artist to the style of another.

In this work, we will explain two architectures used for style transferring: Convolutional Neural Networks and Generative Adversarial Networks. Later, we will introduce two different methodologies that use these architectures in order to generate new samples under the effect of style transfer. Lastly, results of a simple code implementation will be shown and compared qualitatively.

One open issue is the validation of the generated outputs. So far, there are many heuristics that measure the quality of the results in one way or another, prioritizing different features. In future work, we intend to analyze these heuristics and the far possibility of implementing a validation routine that would enable a numerical or analytical comparison of sample quality among different models. In this sense, the following research represents an introduction of the basics to this area of study.

35th Conference on Neural Information Processing Systems (NeurIPS 2021).

2 State of the art

Before explaining the methods used to carry out style transferring, the architectures involved in the experiments should be introduced. To fulfill this goal, the following concepts will be explored: CNN, GAN and CycleGAN[1] (Lorenzo and Xavier).

2.1 Convolutional Neural Networks

Convolutional Neural Networks (CNNs)[2] (Lorenzo) are a type of Artificial Neural Network (ANN) that is specialized in the field of pattern recognition within images. In that sense, the structure of a CNN will be similar to the one of ANNs. A CNN is comprised of an input layer that should be able to handle images represented as multidimensional vectors, a set of hidden layers to which the input is distributed and an output which will be improved in the *process of learning*. We refer as learning to the process of hidden layers adopting stochastic changes in order to improve the final output.

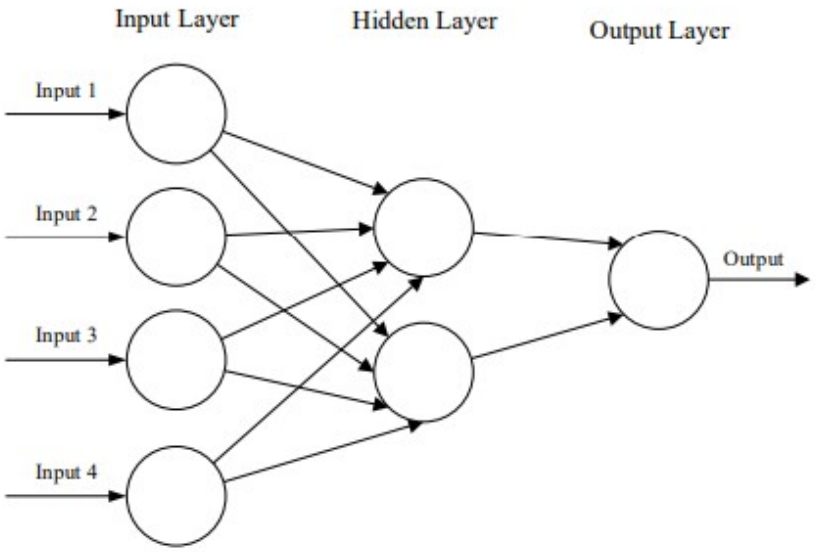

Figure 1: Sketch of a basic ANN architecture.

CNNs are able to encode image-specific features into its architecture while reducing the parameters required to set up the model. The architecture of a CNN is comprised of three kind of layers: convolutional, pooling and fully-connected layers.

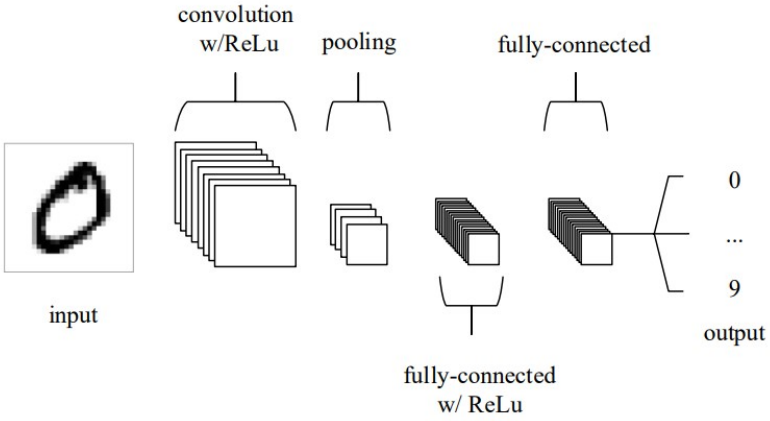

Figure 2: Architecture of a simple 5-layer CNN. Taken from [2]

After the input layer holds all the pixel values of the image, convolutional layers come in play. These layers are a key element in the architecture of a CNN.

Convolutional layers use learnable kernels in order to glide through the input, computing the scalar product between the pooled vector and the weights of the kernel. In this way, the network learns kernels that activate themselves when they detect a specific feature at a given spatial position of the input.

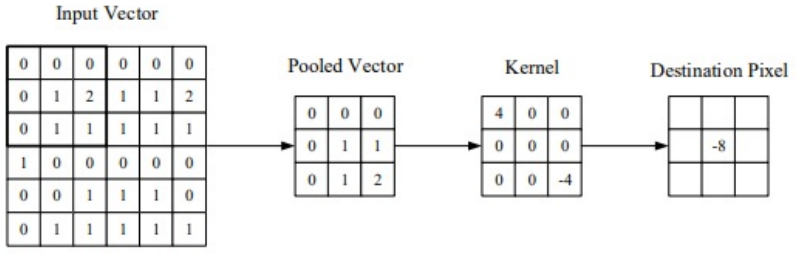

Figure 3: Sample of the behavior of activation kernels in one step of the gliding process. Taken from [2]

It should also be noted that the pooling layers are added in order to reduce the dimensionality of the representations. Finally, fully-connected layers work in the same way as in a conventional ANN structure.

When CNNs are trained on object recognition, the lower layers from the network reproduce the exact pixel values of the original image. However, the higher layers from the network capture the content of the image, in terms of objects and their arrangement. This feature space produced by the higher layers is known as *content representation*.

In order to capture the *style* of the input image, a feature space originally designed to capture texture information should be used. In this space, a representation of the style can be found in the lower layers of the architecture.

Therefore, as mentioned in [3] (Lorenzo and Xavier), the representations of content and style are separable. This phenomena allows us to abstract the style from the input image using CNNs and transfer it to another image to generate an output. Later in this paper, we will explain how this process works and compare it with techniques involving GANs.

2.2 Generative Adversarial Networks

Generative Adversarial Networks (GANs)[4] (Lorenzo) are networks that implicitly model high-dimensional distributions of data. There are two principal components that take part in a GAN: the generator, $\mathcal{G}$, which tries to produce realistic images and the discriminator, $\mathcal{D}$, which aims to tell apart the real images from the generated ones. In our context, it is helpful to consider the following analogy: G is an art forger and D is an art expert trying to identify forgeries.

Both components are trained simultaneously in a sort of competition to be better than the other one. It is important to note that the generator has no access to real images: it learns to generate better outputs only through the interactions with the discriminator.

Usually, the main interest in GANs is not either the discriminator or the generator by itself. The main applications of this kind of architecture focus on the samples that are generated along the way.

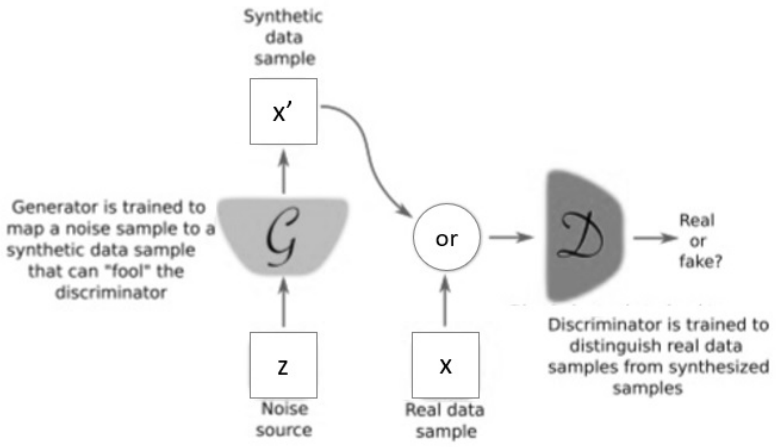

Figure 4: Sketch of a basic GAN architecture.

The simplest architecture of a GAN consists in two Fully Connected Neural Networks (FCNN) working against each other. The key element that models this competition is the two loss functions used in the training of the architecture. These loss functions receive the name of *adversarial losses*.

Taking into account that we are working in the domain of images, substituting the FCNNs for CNNs is a natural extension of this architecture. Nonetheless, in order to compare the results of CNNs and GANs we should consider a different path of extending GANs.

2.3 CycleGANs

Style transfer to an image can be seen as a translation of an image from one style to another. Merely using a GAN to generate indistinguishable samples from a real set of images does not guarantee that the generated samples are matched with the real samples in a meaningful way.

For that reason, we aspire to build an architecture that is able to translate a sample to a different style and then retrieve the original sample once again by translating it back. This characteristic is commonly known as *cycle consistency*.

Imposing cycle consistency is a common technique in the language domain, as translating a sentence back to its original language is an intuitive way to verify the quality of the translations.

In this work, we aim to produce transfer styles onto images using **Cycle Consistent GANs** (Cycle-GANs) and compare its outputs with those generated by CNNs. From now on, we will focus on giving more details on how both methodologies are implemented.

3 Methods

Our goal is to generate a new image x by using a content image c and a style image s. We want to combine the content from c, e.g. a photograph, with the style from s being a painting from a well-known artist, e.g. Van Gogh. Let's introduce the methods used to achieve this goal.

3.1 VGG-Networks (CNNs)

The first method being introduced is a CNN[3] having a subset of layers from the VGG-Network[5] (Xavier): a CNN that rivals human performance on a common visual object recognition task. The feature space used is defined by some of the convolutions from the original VGG-Network: the first convolution layer's outputs from each one of its total 5 blocks for the style representation and the outputs of the second convolution layer from the fifth block for the content representation. Generally, each layer defines a non-linear filter bank so that a given input image is encoded in each layer of the CNN by the filter responses of that image.

Considering our style transfer problem, the following statements are assumed[3]:

- Two images are similar in content if their high-level features as extracted by an image recognition system are close in Euclidean distance. High level features in classifiers tend to correspond to higher levels of abstraction[6].
- Two images are similar in style if their low-level features as extracted by an image recognition system share the same spatial statistics. Artistic style of a painting may be interpreted as a visual texture[7].

Considering x to be the stylized image, c the content image and s the style image, we could define the loss function as[8] (Xavier):

$$\mathcal{L}(x, c, s) = \lambda_c \mathcal{L}_{content}(x, c) + \lambda_s \mathcal{L}_{style}(x, s) \tag{1}$$

where $\mathcal{L}_{content}(x, c)$ and $\mathcal{L}_{style}(x, s)$ are the content and style loss respectively and λ_c and λ_s are scaling hyperparameters weighting the strength of each loss.

Given a set of lower layers $\mathcal{S}$ (the style layers) and a set of higher layers $\mathcal{C}$ (the content layers), assuming the two points above, the content and style losses are defined as:

$$\mathcal{L}_{content}(x, c) = \sum_{i \in \mathcal{C}} \frac{1}{n_i} ||f_i(x) - f_i(c)||_2^2 \tag{2}$$

$$\mathcal{L}_{style}(x, s) = \sum_{j \in \mathcal{S}} \frac{1}{n_j} ||\mathcal{G}(f_j(x)) - \mathcal{G}(f_j(s))||_F^2 \tag{3}$$

where $f_l(x)$ are the outputs from layer l, n_l is the number of units at layer l and $\mathcal{G}(f_l(x))$ is the Gram matrix associated to the layer l outputs.

Finally, to generate the image mixing the content of an image with the style of another one, we jointly minimize the distance from a white noise image from the content representation and the style representation together. That is minimize the expression given in equation 1 using the outputs defined by the CNN architecture.

3.2 CycleGAN

The second method is CycleGAN. This model is an extension of the original GAN architecture that involves the simultaneous training of two GANs, i.e., two generators and two discriminators. One generator takes images from the first domain and outputs images from the second domain. The other generator takes images from the second domain as input and generates images for the first domain. Then, discriminators models determine how plausible the generated images are and update the generator models accordingly. Through this architecture, CycleGAN achieve the desired cycle consistency that has been defined in previous sections. It is worth reminding that the key idea is that an image output by the first generator could be used as input to the second generator and the output of this second should match the original image (the reverse should also be true).

Examining our style transfer problem, in a similar way to the other model, we want to combine the content of one image with the style of another based on matching the Gram matrix statistics of pre-trained deep features. However, the current formulation prioritizes to learn the mapping between two image collections by trying to capture correspondences between higher-level appearance structures.

Formally, the goal is to learn a mapping between two domains X and Y given training samples from each domain, $\{x_i\}_{i=1}^{N}$ and $\{y_j\}_{j=1}^{M}$ respectively. As shown in Figure 5 the model has two mappings $G : X \rightarrow Y$ and $F : Y \rightarrow X$, and two discriminators D_X and D_Y to distinguish images from X to Y and vice versa.

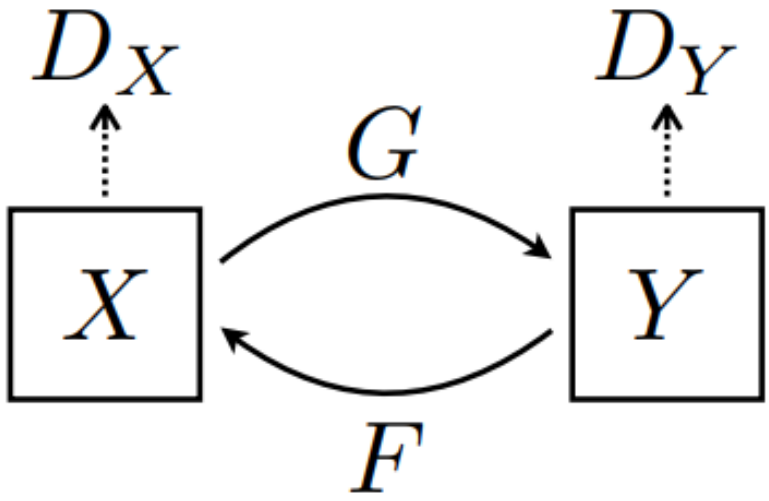

Figure 5: CycleGAN architecture sketch with two mapping functions. Taken from [1]

The objective function consists on two type of terms: *adversarial losses*[9] (Xavier) for matching the distribution of generated images to the data distribution in the target domain; and *cycle consistency losses* to prevent learned mappings G and F from contradicting each other.

Denoting the data distribution as $x \sim p_{data}(x)$ and $y \sim p_{data}(y)$, the adversarial losses are applied to both mapping functions. For the mapping G and discriminator D_Y, the objective is expressed as:

$$\begin{aligned}\mathcal{L}_{GAN}(G, D_Y, X, Y) = \mathbb{E}_{y \sim p_{data}(y)}[\log D_Y(y)] \\ + \mathbb{E}_{x \sim p_{data}(x)}[\log(1 - D_Y(G(x)))].\end{aligned} \tag{4}$$

G aims to minimize this objective against the adversary D that tries to maximize it. Also, an adversarial loss for the mapping F and discriminator D_X is defined in the same way.

The adversarial losses allow the learning of mappings G and F to produce images on the domains Y and X respectively. However, the property that characterizes this model, the cycle consistency, requires to define another loss to ensure that for each $x \in X$, its mapping into Y can be brought back to the original image, i.e. $x \rightarrow G(x) \rightarrow F(G(x)) \approx x$, also called *forward cycle consistency*[1]. In the same way, for each image $y \in Y$, the mapping in X can be brought back to the original image, also called *backward cycle consistency*[1]. This behavior is modelled by the cycle consistency loss defined as follows:

$$\begin{aligned}\mathcal{L}_{cyc}(G, F) = \mathbb{E}_{x \sim p_{data}(x)}[||F(G(x)) - x||_1] \\ + \mathbb{E}_{y \sim p_{data}(y)}[||G(F(y)) - y||_1].\end{aligned} \tag{5}$$

Then, the full objective for this method is:

$$\begin{aligned}\mathcal{L}(G, F, D_X, D_Y) = &\mathcal{L}_{GAN}(G, D_Y, X, Y) \\ &+ \mathcal{L}_{GAN}(F, D_X, Y, X) \\ &+ \lambda \mathcal{L}_{cyc}(G, F)\end{aligned} \tag{6}$$

where λ controls the relative importance of the two objectives.

4 Results

This section exposes some examples generated by the experiments, using the CNN and the CycleGAN approaches. As these methods are currently difficult to evaluate, in terms of quantitative metrics, we are not providing a detailed analysis about the results obtained. However, the results shown will have a brief explanation and also a short comparison between CNN and CycleGAN. The code used to perform the experiments is provided in our github repository[1].

4.1 VGG-Networks (CNNs)

As explained previously the technique combines a content image and a style image to produce a new one. Figure 6 presents an example of the experiment run with the CNN approach. The left image shows a landscape we got from a Kaggle dataset[2], the center image is an Alfred Sisley painting also obtained from a Kaggle dataset[3], and the right image is the combined image after running the algorithm.

(a) Content image (b) Style Image (c) Combined Image

Figure 6: Example from applying the CNN technique. a) A landscape content image, b) An Alfred Sisley painting and c) The combined image between the two previous ones.

From a qualitative perspective, the combined image looks acceptable. It preserves the content from the first image while adapting the texture and colors from the second one. The generated image looks very similar to Alfred Sisley's paintings and it could be considered one of his works.

4.2 CycleGAN

The CycleGAN method works similarly to the CNN approach but, in this case, the method learns the features from a content image's domain and from the domains of several style images. So, the experiment does not show the composition of a content image with a single style image. Instead, it generates an image by taking a content image and a learned style distribution from a set of style images. Figure 7 shows the result from applying the CycleGAN model to the same image used previously. The left image is the landscape content image, the center image represents a distribution of Alfred Sisley's paintings and the right image is the result from applying the Alfred Sisley's style learned to the content image.

[1] https://github.com/xavi2411/ML-Workshop-StyleTransfer

[2] https://www.kaggle.com/arnaud58/landscape-pictures

[3] https://www.kaggle.com/ikarus777/best-artworks-of-all-time

(a) Content image (b) Style Image (c) Stylized Image

Figure 7: Example from applying the CycleGAN technique. a) A landscape content image, b) Some Alfred Sisley paintings and c) The stylized image between the content and the learned style.

Again, it is observable that the generated image preserves the content of the original image and adopts the style learned. We could also say that the generated image imitates Alfred Sisley's style accurately.

In comparison to the previous method, we can detect some differences. The CycleGAN generated image adopts brighter and bluer colors while the CNN one is darker. The reason is the style representation being used: while the CNN only uses one style image, the CycleGAN procedure has been trained on a larger set of style images allowing it to learn a higher set of features that can not be detected by the CNN approach. Despite this difference, both images look similar and they are outstanding results to the point that they could be considered two of Alfred Sisley's works.

5 Conclusions

The aim of this paper has been to explore the Style Transfer problem as well as present two state of the art techniques to address it. Basically, the style transfer problem consists on giving a content image (e.g. a photograph) the style of another image (e.g. a famous painter style). To deal with this problem we introduced a CNN method that using a gradient based algorithm generates the desired image, and a CycleGAN model that uses two generative adversarial networks to translate an image between two different domains. We consider that we got amazing results in the sense that they look very similar to the painting style from the artist used, Alfred Sisley.

For future research, we would like to go deeper into the CycleGAN algorithm, perform an advanced training, fine-tunning the hyperparameters and using different artists. Also, it could be interesting to combine different authors to create new styles. In addition, different techniques to address the style transfer problem could be explored such *DRB-GAN*[10], another method based on GANs.

Also, a more quantitative analysis of the results is desired. This can be achieved throughout already existing heuristics. However, it would be interesting to explore the problem of validation in sample generating models.

References

[1] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *CoRR*, abs/1703.10593, 2017.

[2] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks. *CoRR*, abs/1511.08458, 2015.

[3] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. A neural algorithm of artistic style. *CoRR*, abs/1508.06576, 2015.

[4] Antonia Creswell, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A. Bharath. Generative adversarial networks: An overview. *CoRR*, abs/1710.07035, 2017.

[5] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.

[6] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks, 2013.

[7] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. Texture synthesis using convolutional neural networks, 2015.

[8] Golnaz Ghiasi, Honglak Lee, Manjunath Kudlur, Vincent Dumoulin, and Jonathon Shlens. Exploring the structure of a real-time, arbitrary neural artistic stylization network, 2017.

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

[10] Wenju Xu, Chengjiang Long, Ruisheng Wang, and Guanghui Wang. Drb-gan: A dynamic resblock generative adversarial network for artistic style transfer, 2021.