INFO8010 : Neural style transfer

Maxime Meurisse,[1] **Adrien Schoffeniels**,[2] and **Valentin Vermeylen**[3]

[1] *m.meurisse@student.uliege.be (s161278)*
[2] *adrien.schoffeniels@student.uliege.be (s162843)*
[3] *valentin.vermeylen@student.uliege.be (s162864)*

I. INTRODUCTION

We decided to tackle the subject of *neural style transfer* for this deep learning project. The basic idea of this subject is to *merge the style of one image and the content of another image*, for example applying the style of an oil painting to a photo of an animal.

An example of such an application is shown in Figure 1.

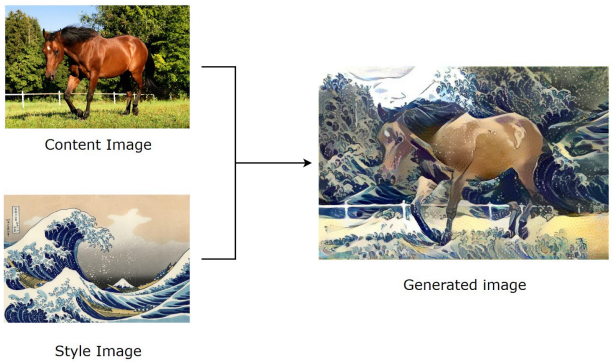

FIG. 1. An example of neural style transfer product [1].

However, the rich domain of neural style transfer does not stop there, many other possible applications and extensions exist : modifying the style of a painting to make it look like a photo (and vice versa), modifying the style of certain elements of a photo (for example, transforming horses into zebras by applying a striped texture to them), transforming a summer landscape into a winter landscape, coloring an image, etc.

The applications are many and varied. We have chosen to focus on the application of merging the style of an image and the content of another image.

To do this, we first started by exploring the technique explained in the paper by Gatys et al. [2]. We tried to re-implement this technique ourselves, to improve it, and to vary the different parameters to understand how it works. This method is described in section III A and its results are discussed in section IV A.

Secondly, we have looked at the CycleGAN (*Cycle-Consistent Generative Adversarial Networks*) technique introduced in the paper by Zhu et al. [3]. This technique, using generative adversarial networks, is very rich and allows many possible applications.

An example of the application of this network of interest for this project is shown in Figure 2.

FIG. 2. An example of neural style transfer produced by CycleGAN [4].

We have also tried to re-implement it by ourselves and to vary its different elements. This method is described in section III B and its results are discussed in section IV B.

Finally, we have tried some extensions in order to delve deeper into the subject : application of neural style transfer without going through a neural network (for comparison), extensions of the application such as controlling the color and adjust the style, etc.

II. RELATED WORK

The field of neural style transfer and its applications are very rich. We have researched the literature to find out what the basic principles of this technique are, what the different techniques are and how they can be improved.

We first based ourselves, as explained in the introduction, on the paper by Gatys et al. and on the paper by Zhu et al. :

- [2] - Paper by Gatys et al. : a paper that introduces a technique for neural style transfer using a network, such as VGG19, already trained.
- [3] - Paper by Zhu et al. : a paper that introduces a technique, called CycleGAN, to perform unpaired

image-to-image translation.

We then found a series of papers highlighting different discussions and improvement techniques :

- [5] - Paper by Gatys et al. : a paper that follows their original one ([2]) and introduces several improvements concerning the results that can be obtained. Examples are the possibility to include bitmaps, to control the colour transfer and the scale.

III. METHODS

A. Neural style transfer (Gatys et al.)

The method presented in this section is that of the original paper that introduced neural style transfer. Although style transfer previously existed in a field called *non-photorealistic rendering* and were implemented through *artistic rendering* algorithms [6], they made no use of neural networks for the task and achieved limited performance and results.

The paper [2] makes use of convolutional neural networks to tackle the task. Those networks are the most powerful when dealing with images since they are able to quickly and efficiently detect local and global correlations in an image. Thanks to filters implemented at each layer, features of the image can be gradually extracted in a feedforward manner, from **smaller to greater details the more we advance or the contrary ?**. The outputs of each layer consist therefore in *feature maps*, which are filtered versions of the input image.

The authors decided to use networks that were pre-trained on object recognition tasks. Such a network creates a representation of the input image that makes the object information increasingly explicit during the treatment, and the content of the image is thus given more weight than the fine details. Higher-level layers contain high-level content while lower-level ones contain low-level details. High-level layers are thus used for content representation. On the other hand, the style is captured through correlations between the filter responses of different layers.

The realization that made neural style transfer possible for the authors is that the style and content of an image are separable in a *convolutional neural network*.

As for the practical method proposed by the paper, they used the pretrained VGG19 network, consisting of 16 convolutional layers and 5 pooling ones (**explain somewhere what those are**). The responses in a layer l being the features maps of that layer, they 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. N_l is the number of distinct filters at layer l and M_l is the size of the feature map (height times width).

The loss that the authors have used in their work is the following one :

$$\mathcal{L}_{total}(p, a, x) = \alpha \mathcal{L}_{content}(p, x) + \beta \mathcal{L}_{style}(a, x) \qquad (1)$$

where p is the photograph (content image), x is the constructed one and a is the artwork (style image). Parameters α and β are simply weighting factors controlling the tradeoff between style and content. The content loss is the mean square error between the response matrix of the content image and that of the constructed image, while the style loss is defined as the weighted sum of the mean square errors between the Gram matrices for the generated image and the style one, taken over all layers considered for the style. The *Gram matrix* is a mathematical element that computes the correlation between the different filter responses for a given layer.

B. CycleGAN (Zhu et al.)

CycleGAN is neural networks that allows to perform *unpaired image-to-image translation*. As described on its official web page, "*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.*" [4]. However, in many cases, paired images are not available. This technique has therefore been designed to support unpaired images.

Paired images are images with a common context and common elements (for example, an image of horse in a field and an image of a zebra in the same position in more or less an identical field) while unpaired images are images that are not really the same (for example, an image of a horse in a field and an image of a zebra, not in the same position, not in the same size and not in a field).

CycleGAN uses a *generative adversarial network*.

1. Generative models

Generative models are models capable of learning how to create data similar to the data they are provided with (training data). Intuitively, a model acting on data (e.g. writing good scientific articles) must have a good internal representation of this data. This internal representation could then be exploited to perform other tasks (e.g. classifying scientific articles).

More practically, a generative model is a model that attempts to learn the joint probability distribution $P(x, y)$

between data x and their label y (where a non-generative model learns the conditional probability $P(y|x)$). Thus, the model would be able to generate a new sample (x, y).

2. Generative adversarial networks

Generative adversarial networks are models whose main idea is to put in competition two neural network models. One is called the *generator* and is in charge of generating noisy data. The other is called the *discriminator* and receives the training data and the noisy data from the generator and has to be able to distinguish between the two sources.

An illustration of the structure of a generative adversarial networks is presented in Figure 3.

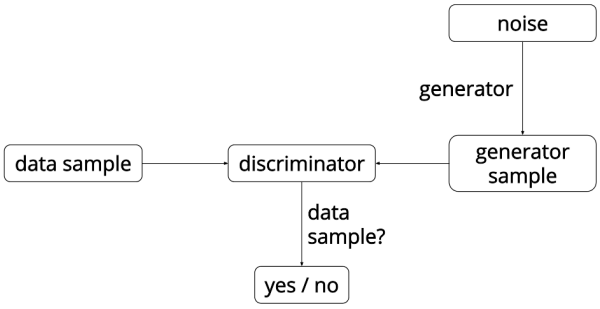

FIG. 3. Generative adversarial networks structure [7].

As the training progresses, the generator produces less and less noisy data and the discriminator becomes better at distinguishing the source from the data it receives. The ultimate goal is to make the generator generate data that will be indistinguishable from the training data by the discriminator.

In such a model, we can backpropagate gradient information from the discriminator back to the generator network, so the generator knows how to adapt its parameters in order to produce output data that can fool the discriminator.

3. CycleGAN

Coming back to the application of neural style transfer, the idea behind the use of a generative adversarial network is *to learn a mapping between the input image and the output image*. However, this technique works for paired images. CycleGAN is a technique that works on unpaired images (or more generally, unpaired input data). For example, it will be able to transform a horse in a photo into a zebra (by applying a striped texture to it) without even needing a zebra image in the same position, with the same background, etc. (i.e. paired images of horse and zebra).

The operating principle of CycleGAN is as follows : let one set of images (for example, images of horses) in an X domain and another different set of images (for example, images of zebras) in a Y domain. The goal is to obtain, via training, a mapping $G : X \to Y$ and a mapping $F : Y \to X$. Mathematically, G and F should be inverse to each other and both mapping should be bijections. This hypothesis being made, G and F are trained simultaneously via a *cycle consistency loss* that encourages $F(G(x)) = x$ and $G(F(y)) = y$.

The creators of CycleGAN have in fact exploited the fact that a translation must be "*cycle consistent*" using this technique. Concretely, this means, for example, that if a sentence is translated from English into French, the re-translation of the freshly obtained French sentence into English should give the original sentence.

Exploiting this property avoids problems such as *mode collapse* : the model could generate an image which is in the domain of Y, and which would therefore be validated by the discriminator, without having any link with the domain X (for example, generating an image of a zebra that is always abd always the same and does not look anything like the image of the original horse).

In total, the model contains two loss functions : an *adversarial loss function* (since it is a generative adversarial model) and a *cycle consistent loss function* (to take into account the cycle consistent property).

The adversarial loss function is defined as

$$\begin{aligned}\mathcal{L}_{\text{GAN}}\left(G, D_Y, X, Y\right) = \mathbb{E}_{y \sim p_{\text{data}}\,(y)}\left[\log D_Y(y)\right] \\ + \mathbb{E}_{x \sim p_{\text{data}}\,(x)}\left[\log\left(1 - D_Y(G(x))\right)\right]\end{aligned} \tag{2}$$

where D_Y is a discriminant which aims to distinguish the images y from the translated images $G(x)$. They also consider the same loss for D_X.

The cycle consistent loss function is defined as

$$\begin{aligned}\mathcal{L}_{\text{cyc}}\left(G, F\right) = \mathbb{E}_{x \sim p_{\text{data}}\,(x)}\left[\|F(G(x)) - x\|_1\right] \\ + \mathbb{E}_{y \sim p_{\text{data}}\,(y)}\left[\|G(F(y)) - y\|_1\right]\end{aligned} \tag{3}$$

taking into account that the result must satisfy the *forward cycle consistency* ($x \to G(x) \to F(G(x)) \approx x$) and the *backward cycle consistency* ($y \to F(y) \to G(F(y)) \approx y$).

An illustration of the cycle consistent loss is shown in Figure 4.

Finally, the total loss function that the model tries to

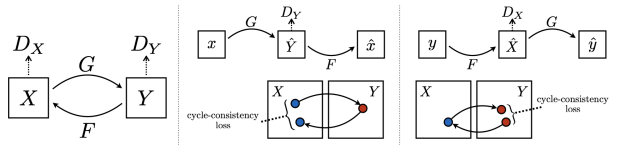

FIG. 4. Cycle consistent loss [3]

minimize is defined as

$$\mathcal{L}\left(G, F, D_X, D_Y\right) = \mathcal{L}_{\text{GAN}}\left(G, D_Y, X, Y\right) + \mathcal{L}_{\text{GAN}}\left(F, D_X, Y, X\right) \quad (4)$$
$$+ \lambda \mathcal{L}_{\text{cyc}}(G, F) \quad (5)$$

where λ controls the relative importance of the two objectives.

4. Network architectures

Concerning generator architectures, authors of CycleGAN use architecture from Johnson et al. [8] : 6 residual blocks for 128×128 training images and 9 residual blocks for 256×256 or higher-resolution training images.

Concerning the discriminator architectures, they use a 70×70 PatchGAN.

5. Implementation

In order to implement the CycleGAN, we decided to not base ourselves upon a tutorial but instead to try to recreate the models and its training from the paper [3] itself. The network architecture is referenced in the fourth section of this paper and makes reference to the network used in another paper [8]. The exact architecture of the networks of this paper being given in the supplementary and using the discriminator architecture of [9]. We therefore tried to implement our network based on these sources and finished doing so before realizing that the exact architectures of CycleGan were given in the appendix of their paper, located after the sources. We thus used that source to modify the networks created to have the same basis as them. For the generator, we did not modify anything since the architecture we used and modified based on their comment in their own implementation was the same as the one provided in their appendix. For the discriminators, however, we had to modify our architecture since they did not mention that the kernel sizes were of 4, and so we went with the kernel size of 3 referenced by the original paper they based upon, and they used a stride of 2 for all layers, which was not the case in [9]. The number of blocks they used also differed.

Finally, after having implemented the network, we went to their GitHub repository for the paper and compared our architecture and theirs, to make sure everything was correct. The only modification we had to make was change some reflection padding to simple padding with 0. They mentioned in their paper they used reflection padding to reduce artifacts but it was not done for all convolutional layers, so we modified that.

After the network architectures, we had to acquire a dataset of images and paintings. To do so, we used a bash script found on their GitHub repository that downloads the same datasets as they used in their research. We decided to use landscapes and Monet paintings for the two domains of interest. We also had to create a custom data loader that would return an image of both domain when called, and to do so we looked into PyTorch documentation and StackOverflow questions.

For the optimizer, we reused the same as the authors, and created a learning rate scheduler based on what they wrote in their paper.

IV. RESULTS

A. Neural style transfer (Gatys et al.)

to do

B. CycleGAN (Zhu et al.)

to do

V. DISCUSSION

to do

[1] Vamshik Shetty. Neural style transfer tutorial -part 1. `https://towardsdatascience.com/neural-style-transfer-tutorial-part-1-f5cd3315fa7f`, 2018.

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

[3] 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.

[4] Jun-Yan Zhu. Unpaired image-to-image translation using cycle-consistent adversarial networks. https://junyanz.github.io/CycleGAN/, 2017.

[5] Leon A. Gatys, Alexander S. Ecker, Matthias Bethge, Aaron Hertzmann, and Eli Shechtman. Controlling perceptual factors in neural style transfer. *CoRR*, abs/1611.07865, 2016.

[6] Yongcheng Jing, Yezhou Yang, Zunlei Feng, Jingwen Ye, and Mingli Song. Neural style transfer: A review. *CoRR*, abs/1705.04058, 2017.

[7] John Glover. An introduction to generative adversarial networks (with code in tensorflow). https://blog.aylien.com/introduction-generative-adversarial-networks-code-tensorfl 2016.

[8] Justin Johnson, Alexandre Alahi, and Fei-Fei Li. Perceptual losses for real-time style transfer and super-resolution. *CoRR*, abs/1603.08155, 2016.

[9] Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew P. Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. Photo-realistic single image super-resolution using a generative adversarial network. *CoRR*, abs/1609.04802, 2016.

[10] Guodong Zhang. Programming assignment 4: Cyclegan. http://www.cs.toronto.edu/~rgrosse/courses/csc321_2018/assignments/a4-handout.pdf, 2018.