Artistic Style Transfer

Final Project Report

ASSAEL Jérémi
CentraleSupélec
jeremi.assael@supelec.fr

AURIAU Vincent
CentraleSupélec
vincent.auriau@supelec.fr

DOYEN Baptiste
CentraleSupélec
baptiste.doyen@supelec.fr

Abstract

The project is split into two parts. The first one aims at developing an algorithm that is able to transfer the style of an image to another. Using two images (called "content" image and "style" image) we will generated a third one using the content of the first one and the style of the second one. This is achieved using an optimization procedure on a loss that is computed using the outputs of intermediate layers of a trained vgg-neural network architecture. This procedure is generalizable for any two images as inputs. In a second time the aim is to create and train a neural network that will be able to stylize the images. It will learn one style and transfer it to any image, way faster than using the optimization procedure. One particularly challenging aspect of this project is to tackle the important computation resources needed to work on images and deep learning architectures. The evaluation part was also a problematic issue as it is difficult to mathematically evaluate the results. Therefore the results will be qualitatively commented.

1. Introduction & Motivation

Some painters are known for a particular painting style (for example Vincent Van Gogh with his brush style). Paintings can also easily be differentiated from photos. We can say that the difference mainly lies in the style of the image. In this project we will try to see how we can modify the style of an image (for example taking a realistic photo and transform it into a cartoon-like picture). More precisely we will try to transfer the style of an image to another one. We can consider that an image is represented by its content and its style, from which we can recreate the exact image. Taking two images as inputs, we want to generate an image with the content of the first one and the style of the second one.
One of the main problem of this project is the mathematical representation if the content and the style of an image. Indeed it is particularly difficult to define an algorithm to extract the style and content of an image. We will later see how we can achieve this.
We will see later that this style transfer is achieved using an optimization procedure. We will try to speed up this process using deep learning. A few questions about the architecture and the needs for a deep learning algorithm will be asked to replace this optimization procedure.

Basically, the questions we want to answer are:

- How can we represent the content and the style of an image?
- How can we transfer the style of an image to another one?
- Is it possible to come up with a fast procedure that will make it usable (almost instant transformation of an image)?

The problem of style transfer in itself is not important, but it can be interesting to be able to extract the content or the style of an image. Indeed in the case of a classifier, extracting the content might give good results if the data pictures have really different styles. We only focus on the core structure of the object which is represented by the content of the image representing this object.

This problem can have a lot of applications. The easiest one is to use it as an infinite source of filters. Indeed on social networks (Snapchat, Instagram, …) filters are particularly popular. Being able to transfer the style of an image gives a lots of interesting possibilities.
We could also imagine new procedures for the realization of cartoons. Indeed using photos and changing their style to a more cartoon-like style, it could create new king animation. A lot of applications of style transger could be imagined. Moreover, art created b AI is growing problematic and this could be something to explore.

2. Problem Definition

We can consider that an image is represented by its content and its style, from which we can recreate the exact image. In this part and the following ones, we will consider three images:
-The style image (S) from which we want to extract the style
-The content image (C) whose style we want to modify and from which we want to extract the content.
-The Generated image (G) that is supposed to have the style of (S) and the content (C) and that basically represents (C) but we the style of (S).

We can write this mathematically as an optimization problem. Indeed we define a loss that we want to minimize:

$$L_{total}(S,C,G) = \alpha L_{content}(C,G) + \beta L_{style}(S,G)$$

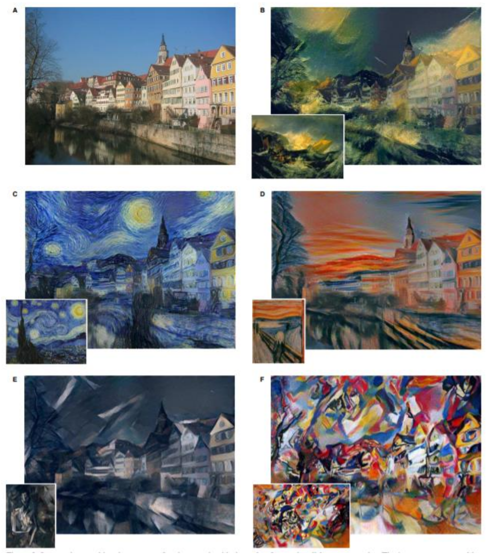
The total loss is constituted of two losses:

- The content loss that evaluates how close the content of (C) and (G) are
- The style loss that evaluates how close the style of (S) and (G) are

One part of this project will be to express this losses in a mathematical way and to create an optimization procedure in order to minimize it. A second part will be to find a method to speed up the stylization process and make it usable.

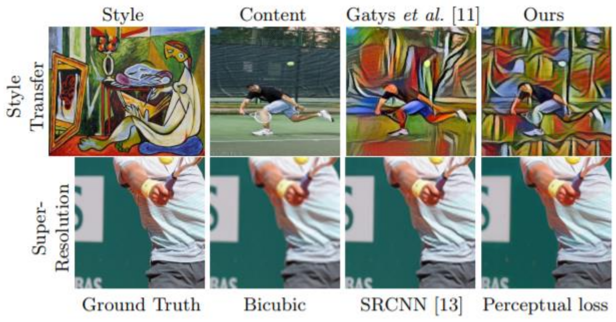
3. Related Work

Leon A. Gatys and al, in their paper *A Neural Algorithm of Artistic Style*, describe their procedure to compute the style and content loss, using a trained VGG-network. They introduced the optimization procedure to transfer style from



one image to another. We can see on the image above an example of their results. We followed this article, in order to develop an algorithm for style transfer. As indicated, we also used a trained VGG-Network in order to compute the losses. However they do not indicate how to optimize the loss, but several classical methods can be used.

J. Johnson and al, in their paper *Perceptual Losses for Real-Time Style Transfer and Super-Resolution*, suggest to use a neural network to learn one style and changed the images on real time. We could not afford the needs (time, example pictures, computation power) to work on the architecture they used, but it inspired us. We tried to train a simpler convolutional network in order to learn one style. This paper was used as a baseline and an inspiration in order to choose the parameters of the network.



We can see on the image above, extracted from this last article that they were able to improve the results from the previous article. They also achieved to train almost perfectly the neural network to stylize on real time the images. We tried to approach this as much as possible.

This project, is also a possibility to let our creativity speak for ourselves with different choices of styles that would make the rendering of our project unique.

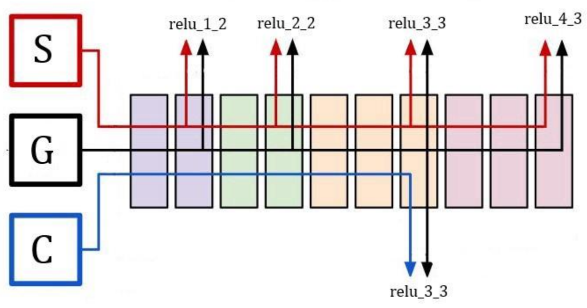
4. Methodology

1. Expression of the Loss

As mentioned earlier, the first problem to tackle is the mathematical expression of both losses, the content loss and the style loss. In their paper, Gatys and all showed that using intermediate layers of deep learning architectures, it was possible to obtain a representation of the content and style of an image. More precisely, they worked on the VGG-Network architecture. Once the network is trained for image classification, it actually learns to separate the content and the style in order to achieve classification. Thus we compare these values for our generated image and our content and style images to compute our losses.

As indicated in the paper, we will also work with a network trained for classification following the VGG19 architecture. The network has been trained to classify 1000 classes and will always be the one used to compute the losses.

The image below illustrates which intermediate outputs are interesting to retrieve the style and the content of an image. The idea is to forward the three images (S, C and G) through the network and retrieve the interesting outputs. For the content we retrieve one ouput and compare the values for the image (C) and (G).For the style, we retrieve four intermediate outputs and compare them for the image (S) and (G). We can now write more mathematically our losses.



Let's first focus on the content. We can call U the layer corresponding to content (relu_3_3 on the image above) and a its activation function. We compare the values at this layers for the content image and the generated image and calculate how close they are using the sum of the square of the difference for all the values:

$$L_{content}(C,\ G) = \frac{1}{2}\sum_{i,\ j}\left(a[U](C)_{i,\ j} - a[U](G)_{i,\ j}\right)^2$$

For the style loss, the same idea is followed, but as we have different outputs with a different number of channels and different sizes of output, we will have to normalize all of the values:

- We will use the Gram matrix of the outputs (instead of just the matrix) which is basically the matrix product of the matrix with its transpose
- For each output we divide by the size of the output

For one style output ($l \in \{$ relu_1_2, relu_2_2, relu_3_3, relu_4_3 $\}$:

$$L_{GM}(S,\ G,\ l) = \frac{1}{4N_l^2M_l^2}\sum_{i,\ j}\left(GM[l](S)_{i,\ j} - GM[l](G)_{i,\ j}\right)^2$$

With GM[l](S) representing the Gram matrix of the output from the layer l for the image S, N_l the number of channels and M_l the $heigth * width$ of this output.
The Gram matrix is just the multiplication of the matrix with its transpose. Using this Gram matrix helps distributing the spatial information contained in the original representation. This is done through the multiplication of each column by a row of the matrix.
Finally the total style los scan be expressed as:

$$L_{style}(S,\ G) = \sum_{l=0}^{V} w_l * L_{GM}(S,\ G,\ l)$$

With w_l a weighting for each ouput, that will basically taken as 1.

Finally the total loss (as seen before) can be written as :

$$L_{total}(S, C, G) = \ \alpha L_{content}(C, G) + \beta L_{style}(S, G)$$

2. Procedure for the generation of one image

Now, we are able to compute our loss and the gradient of this loss toward the pixels of the generated image. We can now apply any optimization algorithm.

We can describe more precisely the process to generate one image (G) from a content image (C) and a style image (S):

First we randomly initialize the image (G)
Then we loop until the results look good enough:

- We forward the three images(S), (C) and (G) through the trained VGG network
- We retrieve the needed outputs from the specific hidden layers
- We compute the total loss using these outputs
- We compute the gradients of this total loss w. r. t. the pixels of (G)
- We apply one step of our optimization algorithm and update (G) (the value of its pixels)

At each step the generated image (G) improves, with its content looking more like the content of (C) and its style to the one of (S).

We can note that the coefficients α and β can be chosen to adapt the result. Each coefficient changes the importance either of the style or the content in the loss and thus modify the result according to this weighting.
In this project they were chosen so that the content of (G) is easily recognizable and the style of (G) with a really different style.
In the use of VGG19, images are resized as inputs and we decided for a size for the generated image which made it quite easy.

3. Learning the stylization with Deep Learning

The stylization using the previous procedure can be evaluated as good after a few minutes and really good after dozens of minutes. In this second part of the project, the objective was to train a neural network so that it learns a style and is able to stylize any image almost instantly.

The previous procedure was able to come up with a generated image with any style and content images. In this procedure, we will learn only one style (taken from one image) and be able to stylize any content image to compute a generated image.

As we wanted to use deep learning to achieve this, we had to create a training dataset. We chose one style from one image and generated a (G) image for one hundred content images (C) using the previous procedure. As it took some time, we limited the previous procedure to around 15 minutes.

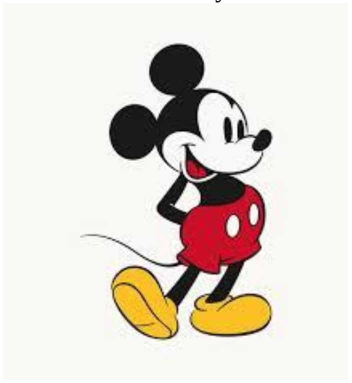
Once the dataset was ready, we could train a neural network. We chose a convolutional network we an architecture of an autoencoder. We limited ourselves to 6 convolutional layers so that the training is not too long. We tried several losses (MSE, binary cross entropy which is used a lost in autoencoder contexts) and used Adagrad as an optimizer.
We trained our network for almost 8 hours and tried different kind of parameters (different number of channels, different sizes of masks, etc…).

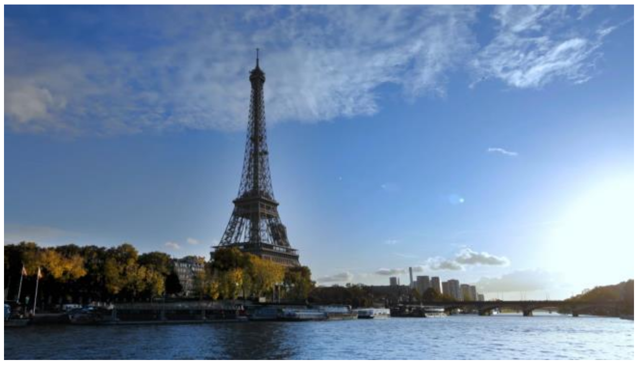
We had to preprocess our data so that it worked well. First, all the images were resized to same size (512x340) and the pixels value were taken between 0 and 1. Finally, we trained the network for many epochs (approximatively 50) because the training set was quiet small.

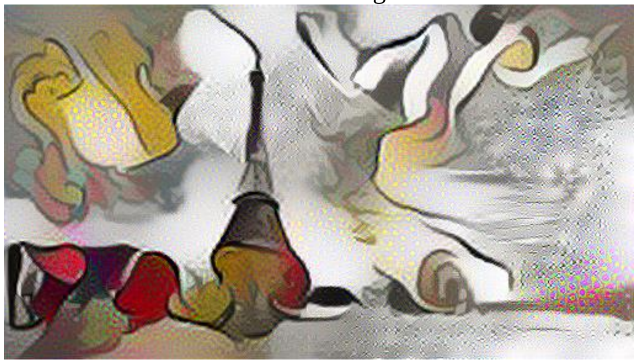
5. Results & Evaluation

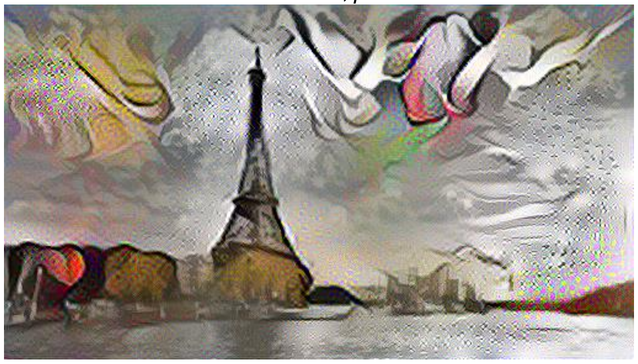
1. Using the Optimization Procedure

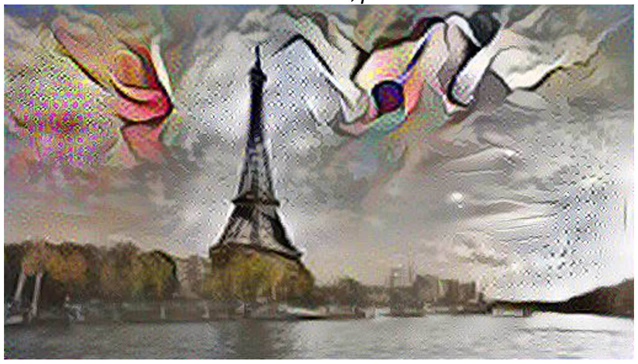
We were able to test different kind of styles using this first procedure. We will present a few of them. A first experiment was conducted with a cartoon image as style and a landscape as the content image. We present three different results with various values for α and β, the weights of the content and loss style.


Style Image


Content Image


Result 1: $\alpha = 1, \beta = 1000$


Result 2: $\alpha = 1, \beta = 100$


Result 3: $\alpha = 1, \beta = 10$

We can quickly see that the bigger β is, the less we can recognize the content of the image. In the third result, we can easily recognize all the trees and the Seine, but it’s more difficult on the first result. We arbitrarily decided that we liked the balance between style and content of the second result and thus the further experiments were conducted using these values for the weights.

The results look pretty good, we kept the content of the original value and we can find the simple lines and the color of the cartoon used as the style image. To be able to run many tests and to keep a certain consistency with the second parts, we decided to run the optimization only on 10.000 iterations (around 15 minutes of computation). For the few tests run on a longer time, the results saw a bigger importance of the style from the Style image.

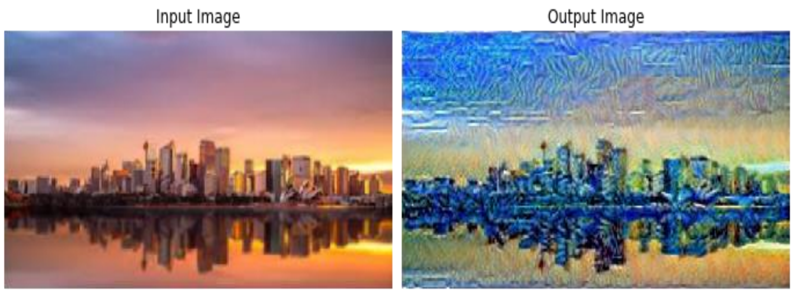
Some pixelation can be observed on some area of the resulting images. A TV loss was added to the two previous ones to minimize this effect. However, this pixilation was a particularity of this style and thus was not found again using other styles. Finally, the TV loss was useless for both this style (because the pixilation was induced by the style loss) and the other styles (because there was not such pixilation) and was given up.

We had not real metric to measure how good our results were. We could only assess that:

- Both loss were indeed decreasing through the optimization
- The generated image's content was the same as the one from the content image and the style really close to the one from the style image.

We concluded that we succeeded in transferring the style from one picture to another.

The famous painting "La Nuit étoilée" from Vincent Van Gogh is used a lot as a style image. Indeed it really represents Van Gogh's brush style and gives a very interesting look to any image. The results obtained (see below) can be compared to the one obtained by Gathis et al. on the first image of this report.

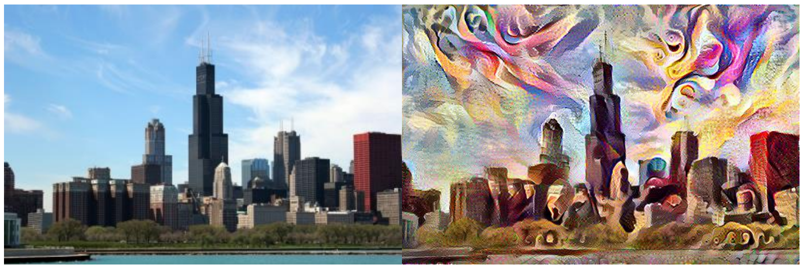


We can easily see that we find the brush style within the output image.

2. Training a neural network for stylization

The first step was to create a dataset of images and their corresponding stylized images. For this we downloaded 100 pictures from Google Image and transformed it using the previous procedure. Below is the Style image and an example of the style transfer with one image of the training dataset. We chose a more colorful style. The time needed to obtain this dataset was already pretty considerable, explaining why we decided to limit ourselves to 10000 iterations (around 15 minutes per image).

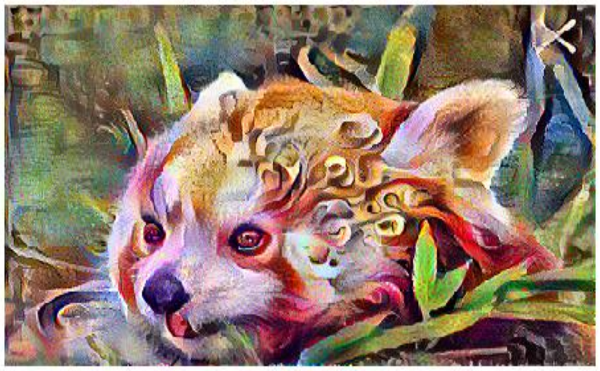
Style Image


Example of the stylization that will be learned

The convolutional network was then trained using this dataset for around 60 epochs (representing around 8 hours of training. Again the results were evaluated in a more qualitatively way than using some metrics. The first thing to observe is that the loss of the neural network was indeed decreasing during the training, being a first hint that the stylization was learned. Even after sixty epochs, the loss was still decreasing. Maybe we there was overfitting and we could have tried to let the network train for a few more epochs, but again we were limited in time and decided not to do it.

Original Image

Below you can find examples of results given for the training set:



Generated Image using first procedure

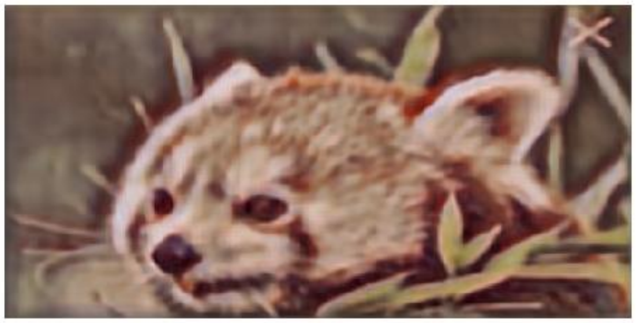


Image generated by the network

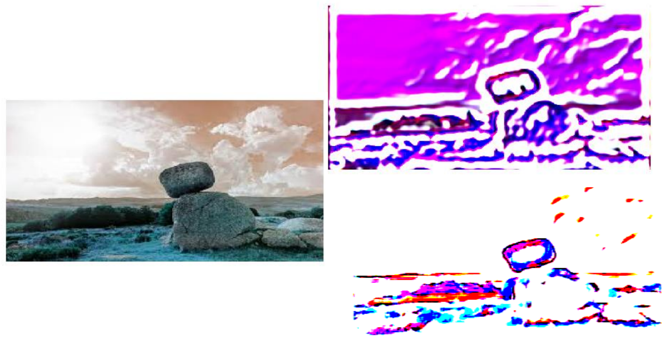
We can see that the network reproduces well the content of the image, however it has difficulties to learn the stylization and the colorization. Indeed the colours are changed but not exactly and the small "whirlwinds" are not drawn. We finally tested it on test images that were not used for the training. The original test image and two images generated by two different architectures

Let's now observe the result for a test picture that was not used for the training. We can observe that the networks keep very well the content of the input image but has difficulties to implement the style. There is a beginning of colorization, but it is quiet far from what we could expect.

We think that these poor results are due to two main factors:

- Frist we did not use an architecture deep enough compared to the one in the article from Johnson et al. We were limited by our computation resources
- Second, we did not have enough training samples. Again we were limited by time and computation power to create enough stylized images.

Major improvements could be achieved if these two obstacles could be overcome.



The original test image and two images generated by two different architectures

It could also be interesting to see if a style with less colours and motives would be easier to learn, but we did not have such an opportunity as we were limited.

6. Conclusions

Through this project we showed that we could get a function that retrieves the content or the style of an image. Convolutional network learn to differentiate these styles and contents of images and we can use it, particularly for style transfer.

Through this project we were able to follow a paper presenting the style transfer and to produce similar results. We succeeded in transferring the style from an image to another. Using the number of iterations or the weights (content and style) we observed that results were different and that the results could be tuned.

We tried to learn the stylization using a deep learning algorithm. However we were pretty limiter in computation power and thus obtained encouraging but not great results. The network keeps the content and tries to apply the stylization but does not succeed really well. As we saw earlier, one intermediate output represent the content and four for the style in the VGG architecture. Thus we can assume that the style is a more complex concept for the network to understand. Thus more complicated and deeper architecture are needed to achieve this task.

The first procedure is greedy in both time and computation power. The second one (deep learning) is faster and easier to use. It could be used on a backend of an app in order to let people stylize any image quickly. However we can think that a deep learning architecture is still a bit heavy in computation and memory to be used easily. An improvement could be to try to make the stylization even faster and in an easier way.

References

[1] L. Gathis, A. Ecker, M. Bethge, *A Neural Algorithm of Style,* 2015.

[2] L. Gathis, A. Ecker, M. Bethge, *Image Style Transfer Using Convolutional Neural Networks,* 2015.

[3] J. Johnson, A. Alahi, F. Li, *Perceptual Losses for real-time style transfer and super-resolution,* 2016