Style Transfer

Paul Bertin*

Mathematiques Vision Apprentissage

E-mail: paul.f.bertin@gmail.com

Abstract

This work is done for the course *Sparsity and Compressed Sensing* taught by Gabriel Peyre in the master *Mathematiques Vision Apprentissage* in ENS Cachan. We do a brief review of the State of the Art for the problem of Style Transfer, mostly applied to images. Then we present our analysis of the unsupervised method from *Frigo et al.*[1] We show that despite its low computational cost compared to other recent methods, this unsupervised method yields very good results. Nevertheless, we show that some assumptions about the content and style images are mandatory.

Introduction

Style Transfer is the problem of transforming an image so that it mimics the style of a given image. In the past, redrawing an image in a particular style required a trained painter and a lot of time. In the last decades, computer scientists have tackled the issue of Style Transfer to produce artificial artworks.

Some methods called *Non-photorealistic Rendering* have first been developed, but they were specific to a given style, and did not generalize. Recently, the power of Convolutional Neural Networks (CNNs) inspired *Gatys et al.* to develop the so called *Neural Style Transfer*. They found that in a CNN, the representations of content and style were separable. Therefore

they used this finding and proposed an algorithm which was not specific to a given style, and produced fantastic artificial artworks shown in figure 1 page 3. The main idea of the algorithm is to start with a random noise image, and change it iteratively until it matches the style representation of the style image and the content representation of the content image.

This work from *Gatys et al.* has been given a lot of attention, and lots of derieved methods have been proposed, to improve either the rendering of the style or the computational efficiency of the transfer. Those works have led to many industrial applications, the *Prisma* mobile application being one of them. The concept of Style Transfer can be extended to other problems such as mimicking the style of a given chess player.[2]

Other methods for texture and style transfer were published recently, which did not require the use of CNN.[1,3] The main idea is to rearrange patches from the style image to create a global *content* which is the one of the content image. Typically, those methods require a much lower computational cost than other ones (for *training* and *prediction* combined). Although those methods differ drasticly from the neural ones, we can find some commonalities.

In this project, we tried to explore both approaches. First, we implemented the *Gatys et al.* method using the GoogLeNet network instead of the VGG one. We used a *Total Variation* regularization during the optimization. Unfortunately, there are lots of parameters to tune, and we did not have a sufficient computational power to achieve reasonable results. We also implemented the method from *Frigo et al.*, tested on various images and found some limitations.

State of the Art : Neural Style Transfer

There are two main classes of Neural Style Transfer algorithms : *Descriptive Neural Methods* Based On Image Iteration and *Generative Neural Methods* Based On Model Iteration.[5]

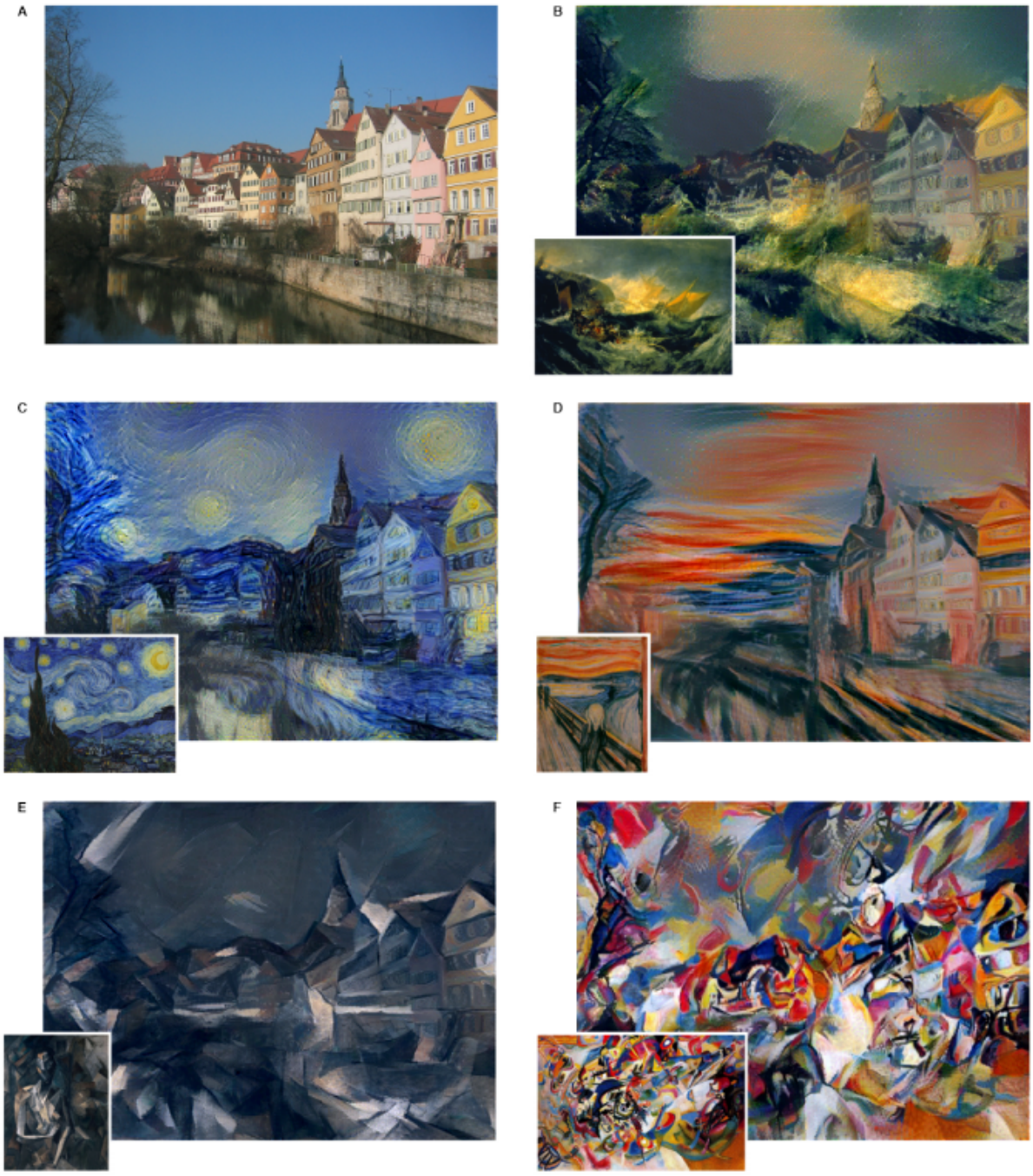

Figure 1: Examples of style transfers using the method from *Gatys et al.*[4]

Descriptive Neural Methods

In *Descriptive Neural Methods*, pixels are updated iteratively in order to match the desired content and style. This optimization problem is given a loss function which we can divide in two terms :

- the content loss E_c which compares the contents of the content image and output image
- the style loss E_s which compares the styles of the style image and output image

One can weight the importance given to each term with parameters α and β.Therefore, given a content image x_c and a style image x_s, the algorithm aims to find a stylized image x which minimizes the total loss :

$$x = \underset{x}{\operatorname{argmin}}\, E_{total} = \underset{x}{\operatorname{argmin}}\, \alpha E_c(x, x_c) + \beta E_s(x, x_s)$$

Typically, E_c and E_s are defined upon the image representations in a CNN. In the original paper from *Gatys et al.*, they are defined as follows :

$$E_c(x, x_c) = \frac{1}{2} \sum_{i,j} ||\Phi^l_{i,j}(x) - \Phi^l_{i,j}(x_c)||^2$$

where $\Phi^l_{i,j}$ is the representation of the i^{th} filter at position j in the layer l of the VGG network.

The style loss uses the Gram matrix $G^l_{i,j}$ to obtain correlations between filter responses. The associated feature space was originally designed to capture texture information. In fact, this is a special case of *Maximum Mean Discrepancy* (MMD) method[6] : matching the Gram matrices is equivalent to minimize the MMD with the second order polynomial kernel.[5]

$$G^l_{i,j} = \sum_k \Phi^l_{i,k} \Phi^l_{j,k}$$

Therefore, given a style representation $G^l_{i,j}$ of the style image in layer l and $A^l_{i,j}$ of the stylized image, we define a loss for the layer l :

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} (G_{i,j}^l - A_{i,j}^l)^2$$

where N_l is the number of filters in layer l and M_l is the size of the feature map in layer l. Finally, we define the style loss :

$$E_s(x, x_s) = \sum_l \omega_l E_l$$

Using different weighting factors ω_l leads to more high or low level Style Transfer. First layers of the network capture low level structures (such as edges etc) whereas deep layers capture high level structures (a whole spiral for instance).

We can add a *Total Variation* denoising term to the loss function. This optimization process requires a lot of computational power and suffers from instabilities : some activations with very different mean and variance can still have the same Gram matrices. To tackle this issue, *Risser et al.* introduced histogram losses.[7]

Even if those methods yield remarkable results, the transfer is not content aware, *i.e.* the content is not taken into account during the transfer process. One could use *Markov Random Fields* (MRF), an approach which we will investigate further when analysing the paper from *Frigo et al.*

Generative Neural Methods

In *Generative Neural Methods*, a generative model is trained, and styled images are produced through a forward pass. As prediction is computationally efficient, those methods are sometimes called *"Fast" Neural Style Transfer*. The key idea is to train a Generator Network over a large set of images for each specific style image.

Johnson et al.[8] proposed to use *Perceptual Loss* to train the Generator. *Perceptual loss* is a loss function based on high level features extracted by a pre-trained neural network. The learning process for this method is several orders of magnitude faster than for the regular

one.

Recently, *Zhu et al.* introduced Cycle-Consistent Generative Adversarial Networks[9] and applied them to style transfer. The goal is to learn a mapping G from a domain X to another domain Y (for instance photo-realistic images and painting images). The key idea of their method is to learn the mapping $G : X \longrightarrow Y$ together with an inverse mapping $F : Y \longrightarrow X$ and enforce $F(G(X)) \approx X$ with a *cycle consistency* loss. Formally, this loss is defined as follows :

$$E_{cyc} = \mathbf{E}_{x \sim p_{data}(x)}[||F(G(x)) - x||] + \mathbf{E}_{y \sim p_{data}(y)}[||G(F(y)) - y||]$$

This loss is used together with the usual Adversarial loss (used to train GANs) which relies on Discrimator networks.

This method yields remarkable results as shown in figure 2 page 6. Moreover, the transfer is *invertible* : one can generate a photo-realistic image from a painting. Note that previous methods obtained poor results for this task.

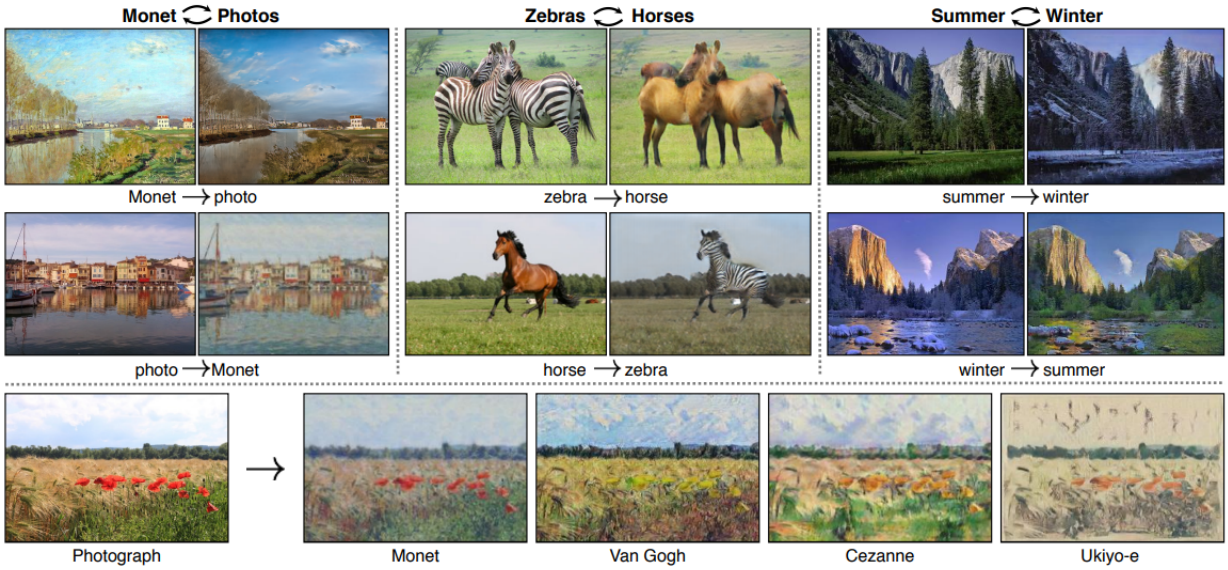

Figure 2: Examples of style transfers using Cycle-Consistent Adversarial Networks

The limitations of those approaches are that they are tied to a specific style, and each new style requires the training of a new Generator Network.

Our approach for Neural Style Transfer

We implemented in Tensorflow the method from *Gatys et al.* and replaced the VGG network by the pre-trained GoogLeNet. We only used the last layer of each Inception module (*i.e.* the layer which merges the activations of the parallel filters of the module). All experiments were led using the *Starry Night* by Van Gogh as style image.

We tried various parameter settings, with and without *Total Variation* regularization but the stylized image always converged to an almost uniformly grey image, as shown in figure 3 page 7. We also tried to initialize directly with the content image instead of random noise.

Note that we were very limited by our computational power, each optimization taking several hours on a personnal laptop.

Figure 3: From left to right : (1) Result of the optimization initialized with random noise (2) Content image from ImageNet (3) Result of the optimization initialized with content image

One of the possible explanations of this behaviour is the instability of the Gram matrix kernel as explained above. The loss function decreased as expected during the optimization, but we do not see any visible structure appearing. This might be because the variance of the distribution of the pixels in the stylized image is too low, even if the stylized image matches activations and Gram matrices in the network.

One could therefore try to use histogram loss as introduced by *Risser et al.*[7] to enforce the pixel distribution in the stylized image to have the same mean and variance as the one in the style image.

Adaptive Patch sampling

We present the method by *Frigo et al.* which is based on an unsupervised re-arrangement of patches from the style image in order to create the content of the content image.

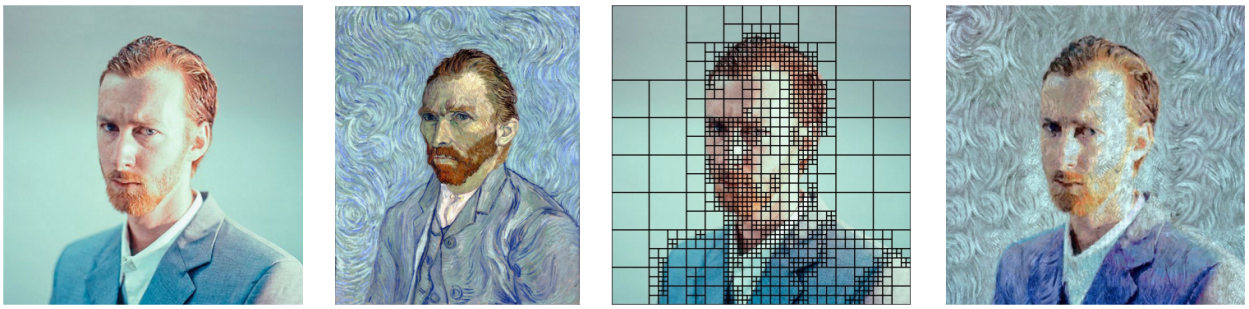

Figure 4: From left to right : (1) Content image (2) Style image (3) Adaptive partition (4) Stylization

The pipeline of the method is the following :

- Split and Match : compute an adaptive partition of the content image
- Search for the optimal map between content patches and style patches
- Bilinear blending between neighbouring regions

We first compute an adaptive patch partition which takes into account both the variance inside the patch of the content image, and its similarity to the style image. The partition has a quadtree structure : if the patch does not match a hand-crafted condition (which aims to predict whether there exist a good candidate style patch for this content patch), it is split into four sub-patches and we iterate. An example partition is shown in figure 4 page 8.

Afterwards, the k-nearest neighbours style patches are computed for each content patch in the partition. A spatial constraint is added so that the neighbours are sufficiently distant from one another in the style image, this is to encourage variety among neighbours. In fact this step is done to lower the complexity of inference in the *Markov Random Field* model.

Then, we compute the most likely assignment of style for all content patches. We want the style patch associated to a given location to be close from the original content patch (fidelity term $\Phi(i)$) but we also want future neighbouring patches i and j to overlap well (pairewise compatibility term $\Psi(i,j)$). More precisely, we want to find the best style labels $L = \{l_i\}$ for the patches i.

Inference in the associated *Markov Random Field* model is done by using the Loopy Propagation algorithm. The factorization of the probability distribution is given by :

$$P(L) = \prod_i \Phi(l_i) \prod_{l_i, l_j} \Psi(l_i, l_j)$$

where

$$\Phi(l_i) = exp(-d[x^i, x_s^{l_i}]\lambda_d)$$

$$\Psi(l_i, l_j) = exp(-d[\tilde{x}_s^{l_i}, \tilde{x}_s^{l_j}]\lambda_s + |l_i - l_j|^2 \lambda_r)$$

with $d[x^i, x_s^{l_i}]$ the distance between patch i in stylized image and its label patch in style image, $d[\tilde{x}_s^{l_i}, \tilde{x}_s^{l_j}]$ the distance over an overlapping region between two style patches i and j which will be neighbours in the stylized image. Note that the term $|l_i - l_j|^2$ encourages two neighbour patches in the stylized image to be far from each other in the style image : this is to avoid spacial repetition. Note that the three parameters λ_d, λ_s and λ_r were set respectively to 2, 2 and 1 in all experiments.

Last, a bilinear blending is done to enforce smooth transitions between patches in the stylized image.

Finally, the authors proposed an optional color and contrast transfer so that one can choose to recover the original color and contrast from the content image. We did not implement this part.

Our approach for Adaptive Patch sampling

We implemented this method using the *factorgraph* library to train the *Markov Random Field* model with the loopy propagtion algorithm. We chosed to focus on this method as it required much less computational power, making experiments easy to do.

For well chosen content and style images, we obtained quite good results as we can see in figure 5 page 10.

Figure 5: From left to right : (1) Stylised image (2) Content image (3) Style image

The effect of the inference through the *Markov Random Field model* is visible in figure 6 page 11. We can clearly see that spatial repetitions are avoided, in particular in the water and sky.

Moreover, We think that *Frigo et al.* forgot to make important assumptions about the style and content images. We need the patches in the style image to be a basis (in some sens) of the content in the content image : the algorithm aims to recreate the content from a spatial combination of style patches. A concrete example of this limitation is the lack of a given color in the style image. As we can see in figure 7 page 12, if a color (here black) is almost absent from the style image, the adaptive partition will be too restrictive, and one will end up with very small patches and a lot of spatial repetition.

One of the possible improvements could be to do the style transfer only on the luminance channel of the image, or to give a higher weight to this channel in the splitting process.

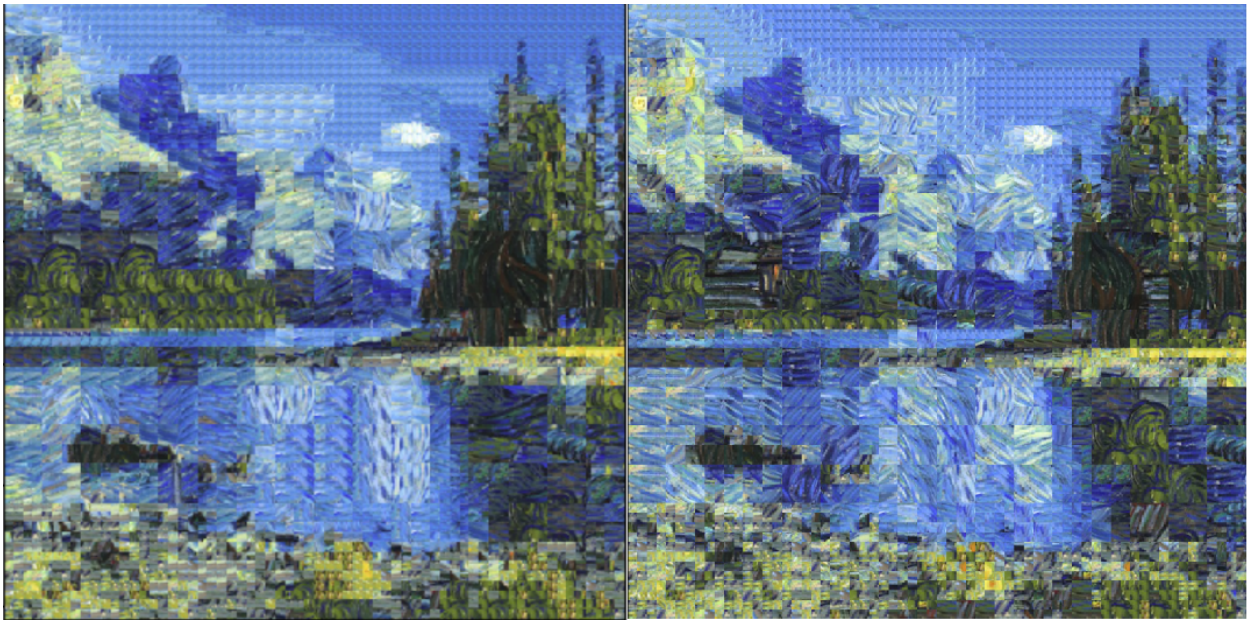

Figure 6: Style transfers before blending. From left to right : (1) Result with nearest neighbours (2) Result with most likely assignment according to the MRF model

Even if this method is interesting and yields quite good results, it is very sensitive to the specificity of the content and style images. Therefore, it is less general than other neural methods which can be applied successfully to a very wide range of content and styles. Nevertheless, the *Content awareness* of the *Frigo et al.* method (*i.e.* here the transfer process is aware of the content) is a very interesting aspect and could be applied to Neural Style Transfer.

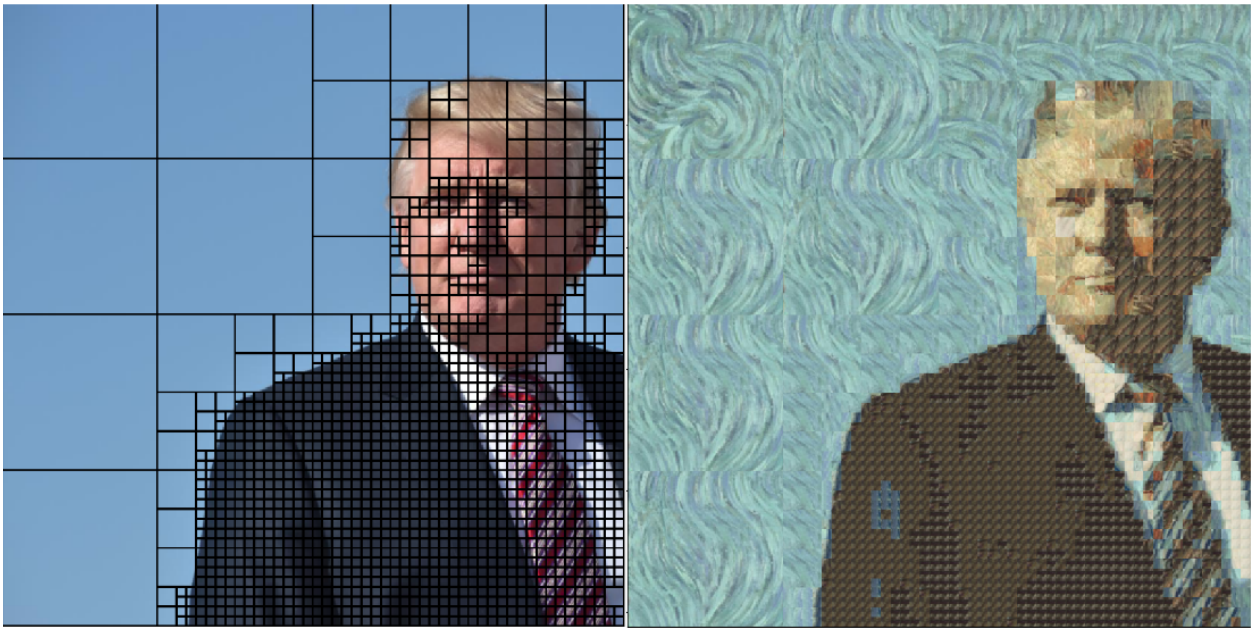

Figure 7: The adaptive partition is too strict in the dark suit region

Conclusion

In this project, we explored a wide range of Style Transfer methods, and were especially interested by Neural methods. Those works are fascinating from a theoretical point of view as they demonstrate content and style can efficiently be separated in some feature space. The adaptive patch sampling method is interesting thanks to its computational efficiency and *content awareness* but lacks robustness. To deal with the robustness issue in this method, one could take the best of both worlds and tried to integrate some deep representation of the images via Neural Networks.

References

(1) Frigo, O.; Sabater, N.; Delon, J.; Hellier, P. **2016**,

(2) Chidambaram, M.; Qi, Y.

(3) Efros, A. A.; Freeman, W. T.

(4) Gatys, L. A.; Ecker, A. S.; Bethge, M. *CoRR* **2015**, *abs/1508.06576*.

(5) Jing, Y.; Yang, Y.; Feng, Z.; Ye, J.; Song, M.

(6) Gretton, A.; Borgwardt, K. M.; Rasch, M. J.

(7) Risser, E.; Wilmot, P.; Barnes, C.

(8) Johnson, J.; Alahi, A.; Fei-Fei, L.

(9) Zhu, J.-Y.; Park, T.; Isola, P.; Efros, A. A.